Calhoun
Institutional Archive of the Naval Postgraduate School

**Calhoun: The NPS Institutional Archive**

**DSpace Repository**

Theses and Dissertations 1. Thesis and Dissertation Collection, all items

1991-12

# A Hopfield network approach to direct adaptive control of nonlinear systems

Starsman, Raymond Scott

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/26576

*Downloaded from NPS Archive: Calhoun*

DUDLEY KNOX LIBRARY

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

http://www.nps.edu/library

# NAVAL POSTGRADUATE SCHOOL
Monterey, California

# THESIS

## A HOPFIELD NETWORK APPROACH TO DIRECT ADAPTIVE CONTROL OF NONLINEAR SYSTEMS

by

Raymond Scott Starsman

December 1991

Thesis Advisor: Roberto Cristi

Approved for public release; distribution is unlimited

T258734

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

## REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*

| | |
|---|---|
| 1a REPORT SECURITY CLASSIFICATION<br>UNCLASSIFIED | 1b RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION/AVAILABILITY OF REPORT<br>Approved for public release; distribution is unlimited |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S) | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |

| | | |
|---|---|---|
| 6a. NAME OF PERFORMING ORGANIZATION<br>Naval Postgraduate School | 6b. OFFICE SYMBOL *(If applicable)*<br>EC | 7a. NAME OF MONITORING ORGANIZATION<br>Naval Postgraduate School |
| 6c. ADDRESS *(City, State, and ZIP Code)*<br>Monterey, CA 93943-5000 | | 7b. ADDRESS *(City, State, and ZIP Code)*<br>Monterey, CA 93943-5000 |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION | 8b. OFFICE SYMBOL *(If applicable)* | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER |
| 8c. ADDRESS *(City, State, and ZIP Code)* | | 10 SOURCE OF FUNDING NUMBERS |

| PROGRAM ELEMENT NO | PROJECT NO | TASK NO | WORK UNIT ACCESSION NO. |
|---|---|---|---|
| | | | |

11. TITLE *(Include Security Classification)* A HOPFIELD NETWORK APPROACH TO DIRECT ADAPTIVE CONTROL OF NONLINEAR SYSTEMS

12. PERSONAL AUTHOR(S)
STARSMAN, Raymond Scott

| 13a. TYPE OF REPORT | 13b TIME COVERED | 14. DATE OF REPORT *(Year, Month, Day)* | 15 PAGE COUNT |
|---|---|---|---|
| Master's Thesis | FROM ________ TO ______ | 1991 December | 108 |

16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the US Government.

17. COSATI CODES

| FIELD | GROUP | SUB-GROUP |
|---|---|---|
| | | |
| | | |

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
direct adaptive control; nonlinear systems; Hopfield net; AUV

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

An automatic control system capable of controlling an unknown nonlinear system is designed using a direct adaptive control scheme, implemented with a Hopfield network. The application of this method to an arbitrary system is discussed in detail and three specific simulation studies are included. These studies include the implementation of the Hopfield network based direct adaptive control system to a linear system, an inverted pendulum, and a nonlinear model of the NPS Autonomous Underwater Vehicle (AUV) with six degrees of freedom. The AUV simulation includes a three dimensional trajectory following algorithm and shows the ability of the Hopfield network to adapt to simultaneous ordered changes in the AUV's depth, speed, and course.

| | | |
|---|---|---|
| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT<br>☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS | 21. ABSTRACT SECURITY CLASSIFICATION<br>UNCLASSIFIED | |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL<br>CRISTI, Roberto | 22b TELEPHONE *(Include Area Code)*<br>408-646-2223 | 22c. OFFICE SYMBOL<br>EC/Cx |

**DD Form 1473, JUN 86** *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603 UNCLASSIFIED

19. cont.

Additionally, an analog circuit design is proposed which implements the automatic control scheme without the support of a microprocessor. The circuit was set up in SPICE and the simulation results as well as some limitations of the analog circuit implementation of the Hopfield network are presented.

Approved for public release; distribution is unlimited.

# A Hopfield Network Approach to Direct Adaptive Control of Nonlinear Systems

by

Raymond Scott Starsman
Lieutenant, United States Navy
B.S.S.E., United States Naval Academy, 1986

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

Thesis
S57775
c.1

# ABSTRACT

An automatic control system capable of controlling an unknown nonlinear system is designed using a direct adaptive control scheme, implemented with a Hopfield network. The application of this method to an arbitrary system is discussed in detail and three specific simulation studies are included. These studies include the implementation of the Hopfield network based direct adaptive control system to a linear system, an inverted pendulum, and a nonlinear model of the NPS Autonomous Underwater Vehicle (AUV) with six degrees of freedom. The AUV simulation includes a three dimensional trajectory following algorithm and shows the ability of the Hopfield network to adapt to simultaneous ordered changes in the AUV's depth, speed, and course.

Additionally, an analog circuit design is proposed which implements the automatic control scheme without the support of a microprocessor. The circuit was set up in SPICE and the simulation results as well as some limitations of the analog circuit implementation of the Hopfield network are presented.

## TABLE OF CONTENTS

I. INTRODUCTION . . . . . . . . . . . . . . . . . . . . . . . . 1

II. ADAPTIVE CONTROL . . . . . . . . . . . . . . . . . . . . . 4

A. PARTIAL STATE REPRESENTATION . . . . . . . . . . 4

B. DIRECT ADAPTIVE CONTROL . . . . . . . . . . . . . . 5

C. LIMITATIONS OF DIRECT ADAPTIVE CONTROL . . . . 9

D. IMPLEMENTATION OF DIRECT ADAPTIVE CONTROL . . . 10

III. THE HOPFIELD NETWORK . . . . . . . . . . . . . . . . 13

A. THE PROCESSING ELEMENT . . . . . . . . . . . . . 13

B. THE HOPFIELD NETWORK . . . . . . . . . . . . . . . 15

C. THE HOPFIELD NETWORK AS A PARAMETER ESTIMATOR . 17

D. THE HOPFIELD NETWORK FOR DIRECT ADAPTIVE CONTROL . . . . . . . . . . . . . . . . . . . . . . 19

E. CONVERGENCE AND STABILITY OF THE HOPFIELD NETWORK . . . . . . . . . . . . . . . . . . . . . . 20

1. The effect of T and C on Hopfield network convergence . . . . . . . . . . . . . . . . . . 21

2. The Excitation of the Input Signal . . . . . 22

F. DIGITAL SIMULATION OF A HOPFIELD NETWORK . . . 23

1. The Processing Element . . . . . . . . . . . 23

2. The Hopfield Network . . . . . . . . . . . . . 23

IV. THE HOPFIELD NETWORK FOR ADAPTIVE CONTROL . . . . . 25

A. THE ALGORITHM . . . . . . . . . . . . . . . . . . . 25

1. Determination of the System Order . . . . . 26

2. Determination of the Reference Model and the Observer . . . . . . . . . . . . . . . . . . . 27

3. Determination of the Weight Filter Pole . . 28

4. Determination of the Input and Output Data Filters . . . . . . . . . . . . . . . . . . . 28

5. Determination of the Control Signal Filter . 29

B. A LINEAR SYSTEM . . . . . . . . . . . . . . . . . 29

C. THE INVERTED PENDULUM . . . . . . . . . . . . . . 34

D. THE AUTONOMOUS UNDERWATER VEHICLE . . . . . . . 38

1. AUV Fundamentals . . . . . . . . . . . . . . 38

2. A Control Scheme for the AUV . . . . . . . . 39

a. The Path Following Algorithm . . . . . . 40

b. A Linear Model of the AUV . . . . . . . 43

(1) The Course Rate Controller . . . . 44

(2) The Depth Rate Controller . . . . . 46

(3) The Speed Controller . . . . . . . 48

c. Summary of Control . . . . . . . . . . . 49

3. Limitations of the Control Scheme . . . . . 50

4. The AUV Control Simulation . . . . . . . . . 50

V. THE HOPFIELD NETWORK AS AN ELECTRONIC CIRCUIT . . . 59

A. A SIMPLIFIED FIRST ORDER SYSTEM CONTROLLER . . 59

B. THE FIRST ORDER SYSTEM IN ANALOG HARDWARE . . . 63

C. THE SPICE SIMULATION . . . . . . . . . . . . . . 64
1. The CMOS Op Amp . . . . . . . . . . . . . . . 64
2. The CMOS Four Quadrant Analog Voltage Multiplier . . . . . . . . . . . . . . . . . . . 65
3. SPICE Simulation of the Hopfield Network . . 67
D. REMARKS ON THE ANALOG CIRCUIT IMPLEMENTATION . 69

VI. SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS . . . . 70
A. SUMMARY . . . . . . . . . . . . . . . . . . . . 70
B. CONCLUSIONS . . . . . . . . . . . . . . . . . . 70
C. RECOMMENDATIONS . . . . . . . . . . . . . . . . 71

APPENDIX A. MATLAB SOFTWARE . . . . . . . . . . . . . . 72

APPENDIX B. TUTSIM CODE . . . . . . . . . . . . . . . 88

APPENDIX C. SPICE SOFTWARE . . . . . . . . . . . . . . 90

LIST OF REFERENCES . . . . . . . . . . . . . . . . . . 94

INITIAL DISTRIBUTION LIST . . . . . . . . . . . . . . . 96

# LIST OF TABLES

Table 2-1 - Direct Adaptive Control Polynomials . . . . 10

Table 4-1 - System Identification of Course Rate Data . 45

Table 4-2 - System Identification of Depth Rate Data . 47

Table 4-3 - System Identification of Speed Data . . . . 49

Table 4-4 - Reference Point Schedule . . . . . . . . . . 52

Table 4-5 - Waypoints for AUV Path . . . . . . . . . . . 56

# LIST OF FIGURES

Figure 2-1 - Partial State Representation . . . . . . . 5

Figure 2-2 - Direct Adaptive Control System . . . . . . 6

Figure 3-1 - A Processing Element . . . . . . . . . . . 13

Figure 3-2 - Sample Transfer Characteristics . . . . . 14

Figure 3-3 - Basic Hopfield Net . . . . . . . . . . . . 15

Figure 4-1 - Hopfield Net Adaptive Controller . . . . . 25

Figure 4-2 - Hopfield control of a Linear System ($g_p$ not limited, $\lambda$=1) . . . . . . . . . . . . . . . . . . . . 31

Figure 4-3 - Hopfield Control of a Linear System ($g_p$ limited, $\lambda$=1) . . . . . . . . . . . . . . . . . . . . 33

Figure 4-4 - Hopfield Control of a Linear System ($g_p$ limited, $\lambda$=1,000,000) . . . . . . . . . . . . . . . . 34

Figure 4-5 - Diagram of an Inverted Pendulum . . . . . 35

Figure 4-6 - Baseline Convergence of the Inverted Pendulum . . . . . . . . . . . . . . . . . . . . . 37

Figure 4-7 - The NPS AUV . . . . . . . . . . . . . . . 39

Figure 4-8 - Posture Definitions . . . . . . . . . . . 41

Figure 4-9 - Course Rate Data and Resultant Models . . 44

Figure 4-10 - Depth Rate Data and Resultant Models . . 46

Figure 4-11 - Speed Data and Resultant Models . . . . . 48

Figure 4-12 - Initial AUV Path Following Simulation . . 53

Figure 4-13 - AUV Control Parameter Vector . . . . . . 54

Figure 4-14 - Improved AUV Path Following Simulation . 55

Figure 4-15 - AUV Traversal of Waypoint Path . . . . . 57

Figure 4-16 - AUV Traversal of Waypoint Path, Following Distance Six Feet . . . . . . . . . . . . . . . . . . . 58

Figure 5-1 - First Order Direct Adaptive Hopfield Controller . . . . . . . . . . . . . . . . . . . . . 62

Figure 5-2 - Results of First Order Hopfield Net . . . 63

Figure 5-3 - CMOS Analog Voltage Multiplier . . . . . . 65

Figure 5-4 - SPICE Hopfield Net Output . . . . . . . . 68

# I. INTRODUCTION

Apart from a few particular cases, no general theory exists for the control of nonlinear systems. The simplest method for the development of a control system for a nonlinear plant is to linearize the nonlinear plant around an operating point and derive a linear controller with classical control methods. These static control algorithms can control nonlinear systems at specific operating points, but may be unstable at others. Simple nonlinear systems such as the inverted pendulum may have stable linearizations at some operating points and be unstable at others. These facts make the control of nonlinear systems difficult.

Adaptive control algorithms hold promise in the control of nonlinear systems. Because an adaptive control system changes in response to changes in the system, it is able to control many nonlinear plants. This thesis investigates the use of a Hopfield net for direct adaptive control of a nonlinear system.

An interesting application is the adaptive control of an Autonomous Underwater Vehicle (AUV). An AUV is an unmanned submersible vehicle designed to operate independently of human interaction or support. As such, it must be capable of responding to changing, dangerous, or unpredictable conditions much as a manned underwater vehicle would. While many of the

AUV's higher level functions, such as path planning and obstacle avoidance, use artificial intelligence techniques to cope with different scenarios, the lower level control remains unsolved.

The dynamics of the AUV are highly non-linear and are not easily rendered into a satisfactory linear form. These non-linearities are particularly evident as the vehicle changes speeds. Schwartz investigated the use of recursive least squares (RLS) and an adaptive pole placement scheme to control the AUV at a given speed [Ref. 1:p. 63]. Although this method produced functional results, the intensive calculations required by the RLS algorithm would further burden the already heavily loaded microprocessor. Since robustness to changing plant parameters (due to changing environmental conditions or damage) is required, an adaptive controller must be used. As the processor aboard is already occupied by the path planning, sensor data processing, and navigation software, a solution to the nonlinear automatic control problem was sought that produces a controller that demands less processor time.

Neural networks offer a potential solution to this problem as there is great promise in the implementation of neural nets in analog hardware. The Hopfield network in particular is easily realized in analog circuitry. An adaptive controller designed using a Hopfield network realized in analog hardware would not overload the on-board processor and yet would provide the necessary robustness for the control of the AUV.

The goal of this thesis is to develop an adaptive control algorithm based on the Hopfield network and to propose a design in analog hardware.

This thesis is organized in five sections. Chapter II describes the direct adaptive control algorithm used throughout this thesis. Chapter III contains a description of the Hopfield network and its application to direct adaptive control. Chapter IV consists of three studies of the implementation of the Hopfield network based direct adaptive control scheme. Chapter V describes a possible analog circuit implementation of this control scheme and discusses some of the problems associated with it. Chapter VI provides a summary of the results of this work and points out several areas for further study.

# II. ADAPTIVE CONTROL

Adaptive control is a method by which a controller is adapted to control an uncertain system in a dynamic operating environment. Two major classes of adaptive control exist: direct and indirect adaptive control. Direct adaptive control is characterized by the direct determination of the control parameters from input and output data collected from the system. Indirect adaptive control is a two stage process. First, system identification techniques are used to obtain a model of the system and then standard control techniques are used to calculate a controller for the estimated model [Ref. 2:p. 48]. Indirect adaptive control is generally slower than direct adaptive control and requires greater hardware support and/or computational effort. In this thesis only direct adaptive control is considered.

## A. PARTIAL STATE REPRESENTATION

In order to proceed with the derivation of the direct adaptive control algorithm, it is necessary to introduce an alternate representation of a system called the *partial state* representation [Ref 3:p. 209]. The system

$$y(t) = \frac{B(s)}{A(s)}u(t) \tag{2-1}$$

where $s$ can be interpreted either as the differential operator or the complex variable of the Laplace transform and

$$A(s) = s^n+a_1s^{n-1}+\ldots+a_n; \quad B(s)=s^m+b_1s^{m-1}+\ldots+b_m \qquad \textbf{(2-2)}$$

can be broken into two components as shown in Figure 2-1.

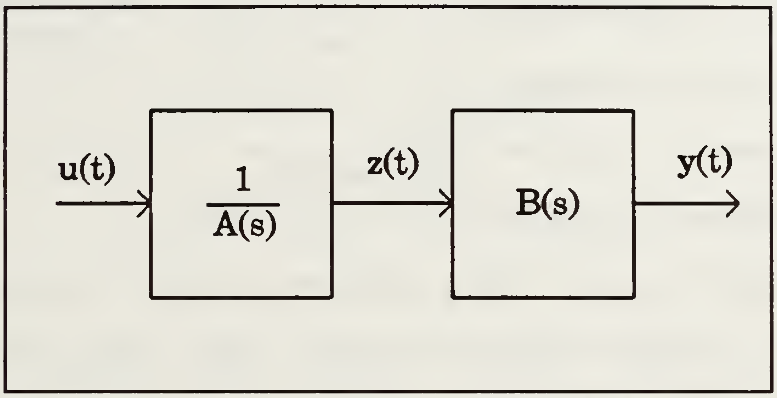

**Figure 2-1** - Partial State Representation

The intermediate state *z(t)* is called the partial state and the system of equation (2-1) is equivalent to

$$\begin{aligned} A(s)z(t) &= u(t) \\ y(t) &= B(s)z(t) \end{aligned} \qquad \textbf{(2-3)}$$

The partial state representation is useful in the derivation of the direct adaptive control algorithm.

## B. DIRECT ADAPTIVE CONTROL

As stated earlier, direct adaptive control uses the input signal to a system and the output signal from a system to directly determine suitable control parameters. For this work a pole placement scheme is employed, meaning that the system output *y(t)* is controlled to react to the reference signal *v(t)* as does the reference model with the transfer function $1/p^*(s)$. A block diagram of a direct adaptive control system is shown in Figure 2-2. In this diagram, the control parameter identifier receives data from the input and output of the system and uses this data to modify the controller.

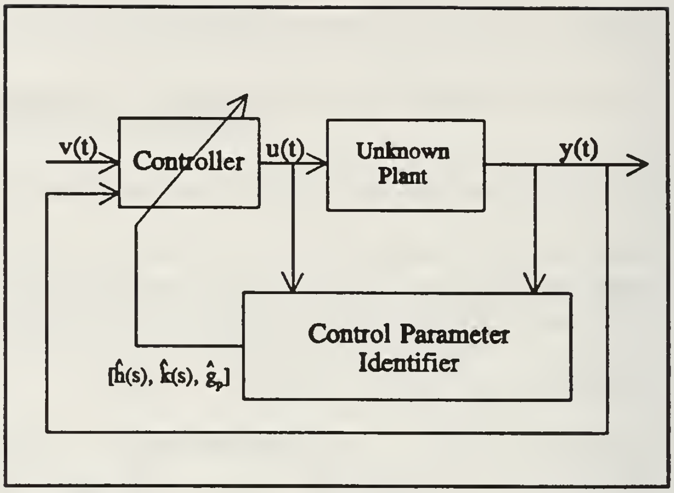
v(t)
Controller
u(t)
Unknown Plant
y(t)
Control Parameter Identifier
$[\hat{h}(s), \hat{k}(s), \hat{g}_p]$

**Figure 2-2** - Direct Adaptive Control System

Assuming the unknown plant can be modeled as a piecewise linear system, then it may be described at any given operating point by the linear differential equation:

$$y(t) = \frac{r(s)}{p(s)}u(t) \tag{2-4}$$

where *p(s)* is an $N^{th}$ order monic polynomial and *r(s)* is an $M^{th}$ order stable polynomial, meaning that the roots of *r(s)* are all in the left half-plane. Rewriting equation (2-4) in partial state form yields:

$$\begin{aligned} p(s)z(t) &= u(t) \\ y(t) &= r(s)z(t). \end{aligned} \tag{2-5}$$

The goal of the controller is to track the output of a reference model driven by an external input $v(t)$, specifically

$$y_m(t) = \frac{1}{p^*(s)}v(t) \tag{2-6}$$

where $p^*(s)$ is the characteristic polynomial of the reference model and $y_m(t)$ is the reference model output. From the pole placement problem the control input has the following structure

$$u(t) = \frac{h(s)}{q(s)}y(t)+\frac{k(s)}{q(s)}u(t)+g_p\cdot v(t) \tag{2-7}$$

where the observer polynomial $q(s)$ is an arbitrary $N^{th}$ order stable monic polynomial, $h(s)$ and $k(s)$ are the unknown control polynomials of order $N-1$, and $g_p$ is the input gain [Ref 4:p. 5]. Multiplying both sides of equation (2-7) by $q(s)$ and substituting for $y(t)$ and $u(t)$ from the partial state equations (2-5) yields the closed loop dynamics

$$\begin{aligned} q(s)p(s)z(t) = h(s)r(s)z(t)+k(s)p(s)z(t)+ \\ g_pq(s)v(t). \end{aligned} \tag{2-8}$$

The polynomials $h(s)$ and $k(s)$ are defined to satisfy the following condition:

$$q(s)p(s)-k(s)p(s)-h(s)r(s) = \frac{1}{r_1}p^*(s)r(s)q(s). \quad \textbf{(2-9)}$$

Equation (2-9) can be put into the form of the Diophantine equation [Ref. 2:p. 291] after some simple rearrangement:

$$h(s)r(s)+k(s)p(s) = q(s)(p(s)-\frac{1}{r_1}p^*(s)r(s)) \quad \textbf{(2-10)}$$

where $r_1$ is the coefficient of the highest order term of the plant numerator polynomial. If the system ($p(s)$ and $r(s)$) were known, then the polynomials $h(s)$ and $k(s)$ could be solved for directly using the Sylvester matrix [Ref. 2:p. 295]. The MATLAB subroutine FIND_HK.M was written to solve the Diophantine equation and return a solution for $h(s)$, $k(s)$, and $g_p$. This subroutine can also be used to determine initial estimates of the coefficients of $h(s)$, $k(s)$, and $g_p$ for a linearized model of a nonlinear system. The subroutine is included in Appendix A.

Assuming the estimates of $h(s)$ and $k(s)$ converge to the solution of the Diophantine equation, then the closed loop partial state equations of the controlled system become:

$$\frac{1}{r_1}p^*(s)q(s)r(s)z(t) = g_p q(s)v(t) \tag{2-11a}$$

$$y(t) = r(s)z(t). \tag{2-11b}$$

Eliminating the partial state variable, *z(t)*, setting $g_p$ to $1/r_1$, and dividing both sides of equation (2-11a) by *q(s)* yields the desired closed loop dynamics:

$$y(t) = \frac{1}{p^*(s)}v(t). \tag{2-12}$$

This transfer function is identical to equation (2-6) and thus the closed loop system now responds to the input *v(t)* as would the reference model given by $p^*(s)$. The challenge remains to find the polynomials *h(s)* and *k(s)* and the gain $g_p$ that satisfy equation (2-9) given only the input and output data of a system.

## C. LIMITATIONS OF DIRECT ADAPTIVE CONTROL

The preceding formulation of a direct adaptive control scheme required several assumptions which are summarized in Table 2-1. Of note is that the unknown plant need not be stable, but must be minimum phase. Because *r(s)*, the unknown plant numerator polynomial, is essentially canceled out by the denominator of the controller, any unstable roots of *r(s)* make the closed loop system internally unstable [Ref. 2:p. 440]. It should also be noted that a good estimate of the number of poles *N* and zeros *M* of the unknown system must be known in order to choose properly sized *q(s)* and *p*(s)* polynomials.

| Polynomial | Order | Form | Must be stable? |
|---|---|---|---|
| p(s) | N | $s^N+p_1s^{N-1}+\dots p_N$ | No |
| r(s) | M | $r_1s^{M-1}+\dots r_M$ | Yes |
| $p^*(s)$ | N-M | $s^{N-M}+p^*_1s^{N-M-1}+\dots p^*_{N-M}$ | Yes |
| q(s) | N | $s^N+q_1s^{N-1}+\dots q_N$ | Yes |
| h(s) | N-1 | $h_1s^{N-1}+\dots h_N$ | No |
| k(s) | N-1 | $k_1s^{N-1}+\dots k_N$ | No |

**Table 2-1** - Direct Adaptive Control Polynomials

## D. IMPLEMENTATION OF DIRECT ADAPTIVE CONTROL

The traditional method of implementing the above direct adaptive control scheme is to use a recursive least squares algortihm to estimate the parameters *h(s)*, *k(s)*, and $g_p$ [Ref. 2:p. 440]. To put this problem into regression form, both sides of equation (2-7) are multiplied by *q(s)* yielding:

$$q(s)u(t) = h(s)y(t)+k(s)u(t)+g_pq(s)v(t). \quad \textbf{(2-13)}$$

Equation (2-12) can be rewritten as:

$$v(t) = p^*(s)y(t). \quad \textbf{(2-14)}$$

Substituting for *v(t)* from equation (2-14) in equation (2-13) yields

$$q(s)u(t) = h(s)y(t)+k(s)u(t)+g_p q(s)p^*(s)y(t). \qquad \textbf{(2-15)}$$

Equation (2-15) can now be written in regression form

$$x(t) = \theta^T \phi(t) \qquad \textbf{(2-16)}$$

where

$$x(t) = q(s)u(t); \quad \theta = \begin{bmatrix} h_1 \\ \vdots \\ h_N \\ k_1 \\ \vdots \\ k_N \\ g_p \end{bmatrix}; \quad \phi(t) = \begin{bmatrix} s^{N-1}y(t) \\ \vdots \\ y(t) \\ s^{N-1}u(t) \\ \vdots \\ u(t) \\ p^*(s)q(s)y(t) \end{bmatrix}. \qquad \textbf{(2-17)}$$

Since $x(t)$ and $\phi(t)$ are known, the least squares estimate of $\theta$ is expressed as [Ref. 5: p. 324]

$$\hat{\theta} = \underset{\theta \in \mathbb{R}^{2N+1}}{\mathit{argmin}} \left[ J(\theta) \right] \qquad \textbf{(2-18)}$$

where

$$J(\theta) = \int_0^t e^{-\alpha(t-\tau)} [x(\tau)-\theta^T\phi(\tau)]^2 d\tau. \qquad \textbf{(2-19)}$$

The value for $\hat{\theta}$ is recursively estimated as

$$\hat{\theta}(kT_s+T_s) = \hat{\theta}(kT_s)+\frac{P(kT_s)\phi(kT_s)[x(kT_s)-\phi^T(kT_s)\hat{\theta}(kT_s)]}{1+\phi^T(kT_s)P(kT_s)\phi(kT_s)}$$

$$P(kT_s+T_s) = P(kT_s)-\frac{P(kT_s)\phi(kT_s)\phi^T(kT_s)P(kT_s)}{1+\phi^T(kT_s)P(kT_s)\phi(kT_s)} \tag{2-20}$$

based on samples taken at a rate of $1/T_s$ samples per second [Ref. 5:p. 325]. Although the RLS method converges to a correct estimate of $\theta$, it requires extensive and intensive calculations, consisting of several matrix multiplications and scalar divisions at every iteration. Furthermore, this process would be difficult to implement in analog hardware because of the number and the nature of the required operations.

# III. THE HOPFIELD NETWORK

## A. THE PROCESSING ELEMENT

The processing element is the heart of any neural network and was conceived as a coarse analogy to the biological neuron [Ref. 6:p. NC-4]. A diagram of a typical processing element is shown in Figure 3-1.

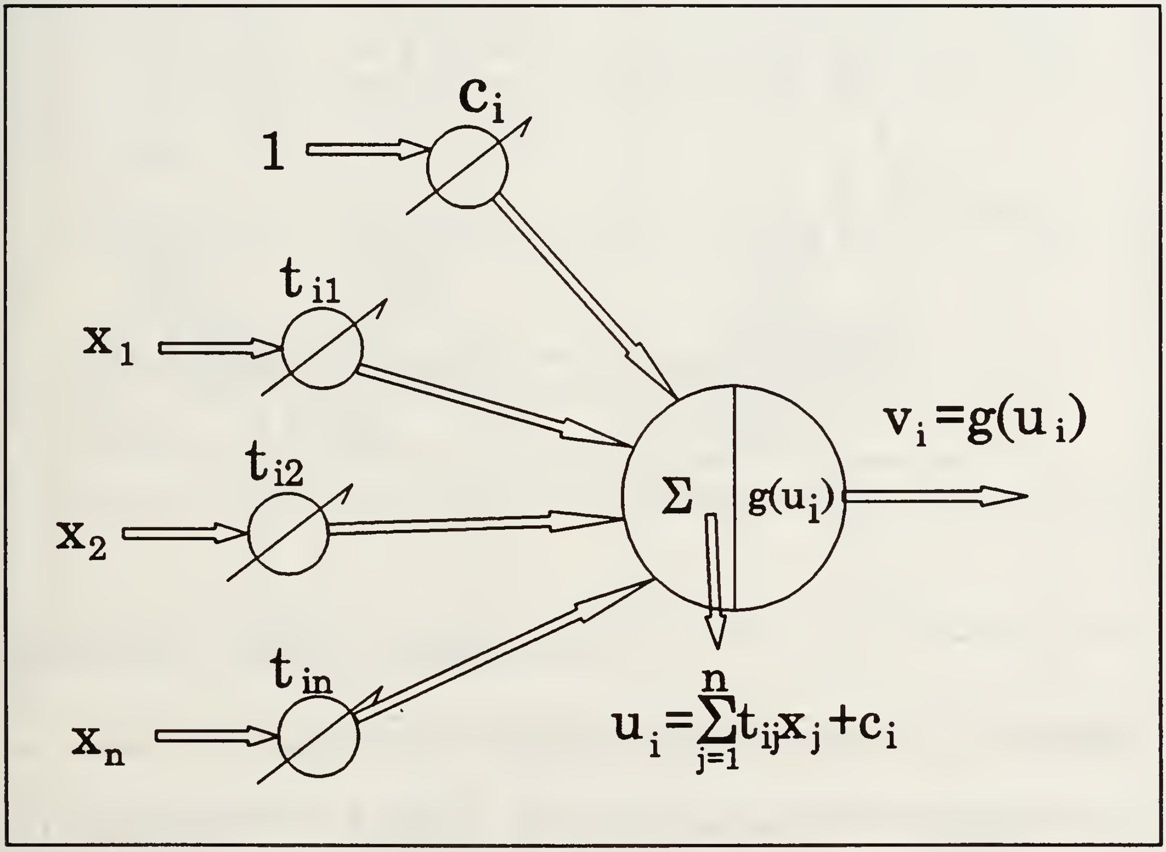

**Figure 3-1** - A Processing Element

The processing element consists of three major parts: the input weights $t_{ij}$; the summing junction; and the transfer characteristic $g(\bullet)$. The inputs $x_i$ are either external signals or signals from other processing elements. The neural network is defined by its structure and the values of the weights. The summing junction simply sums the weighted inputs as well as the weighted bias signal $c_i$ and passes the result $u_i$ to the transfer characteristic [Ref. 6:p. NC-5]. Figure 3-2 shows several samples of possible transfer characteristics.

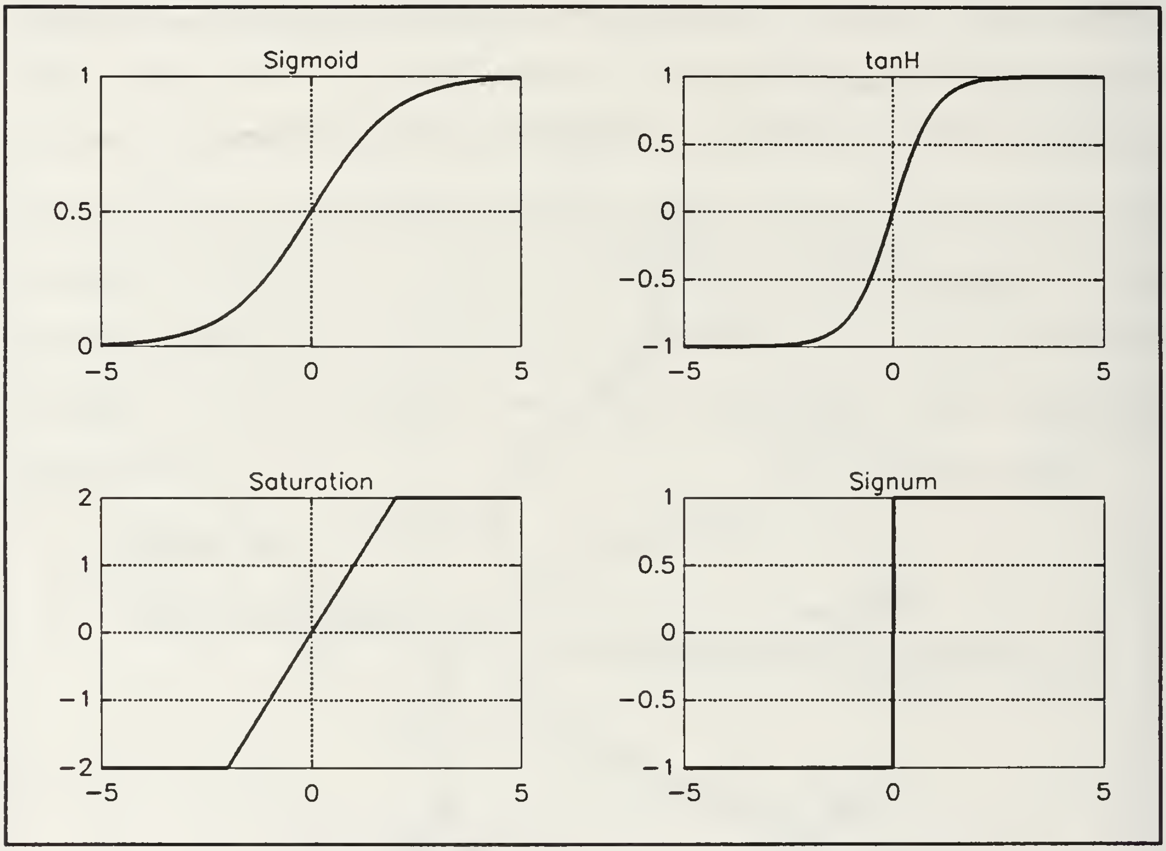

**Figure 3-2** - Sample Transfer Characteristics

The transfer characteristic is usually a monotonically increasing nonlinear function such as a sigmoid or a hyperbolic tangent [Ref 7: p. 40].

Neural networks are typically constructed by arranging processing elements in layers and interconnecting the layers.

## B. THE HOPFIELD NETWORK

The Hopfield network consists of a single layer of processing elements that are completely mutually interconnected [Ref. 8:pp. 96-99]. A generic Hopfield network is shown in Figure 3-3. The Hopfield network uses a version of the processing element introduced in the previous section modified by the addition of an integrator at the output of each neuron. Thus the $i^{th}$ neuron in a continuous Hopfield

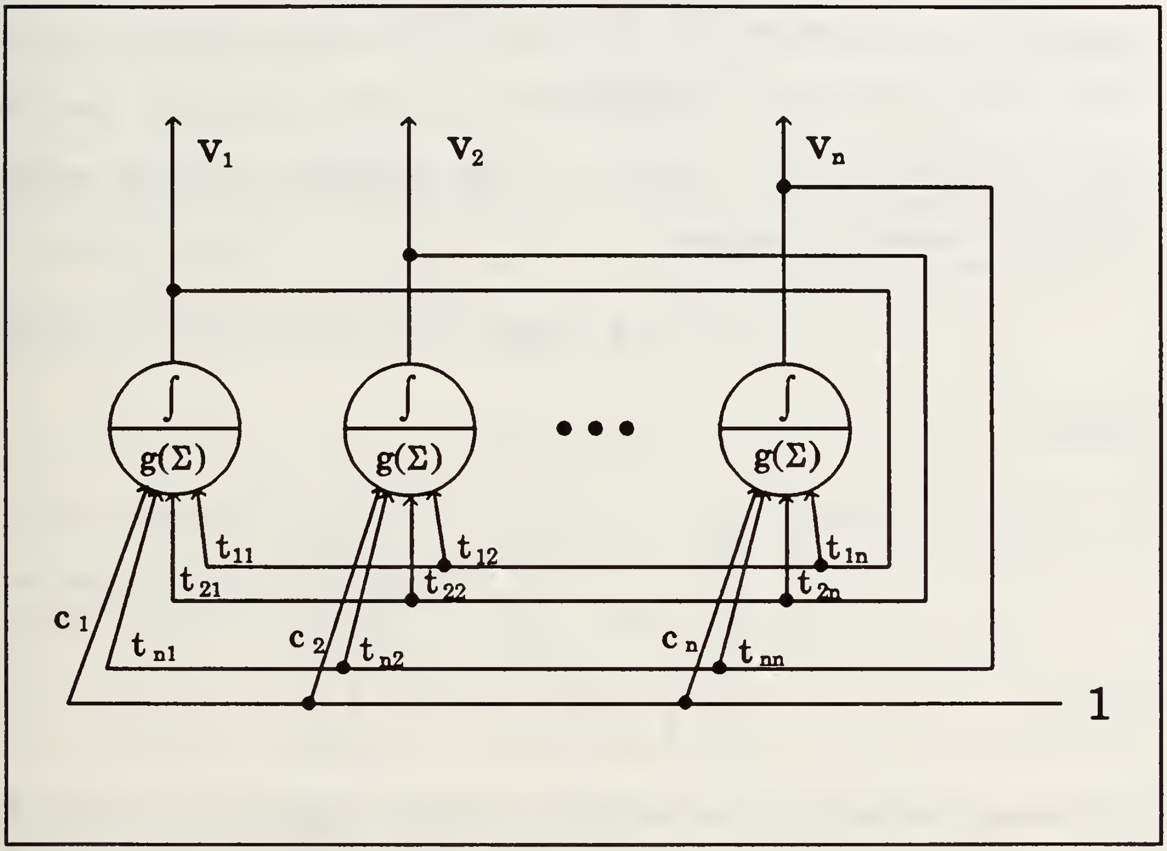

**Figure 3-3** - Basic Hopfield Net

network of $n$ elements evolves as:

$$\dot{u}_i = \sum_{j=1}^{n} t_{ij} v_j + c_i \tag{3-1}$$

where $v_i = g(u_i)$, $g(\bullet)$ being the monotonically increasing transfer characteristic. Arranging the neuron states $v_i$ into vector $\boldsymbol{V}$, the weight coefficients $t_{ij}$ into matrix $\boldsymbol{T}$, and the bias weights $c_i$ into vector $\boldsymbol{C}$, the equation for the entire Hopfield network becomes

$$\dot{U} = TV + C \tag{3-2}$$

where

$$T = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1n} \\ t_{21} & t_{22} & \cdots & t_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ t_{n1} & t_{n2} & \cdots & t_{nn} \end{bmatrix}; \qquad C = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}. \tag{3-3}$$

An energy function is defined for the Hopfield network of equation (3-2) as

$$E(t) = -\frac{1}{2} V^T T V - C^T V \tag{3-4}$$

Hopfield has shown that if $\boldsymbol{T}$ is negative definite and symmetric, then the energy function, $E(t)$, tends to a minimum [Ref. 8: p. 99]. This is shown by taking the time derivative of equation (3-4) along the trajectory of the network:

$$\dot{E}(t) = -\frac{1}{2}\dot{V}^T TV - \frac{1}{2}V^T T\dot{V} - C^T\dot{V}. \tag{3-5}$$

Since $T$ is symmetric, equation (3-5) is simplified to:

$$\begin{aligned} \dot{E}(t) &= -V^T T\dot{V} - C^T\dot{V} \\ &= -(V^T T + C^T)\dot{V} \\ &= -\dot{V}^T(TV + C). \end{aligned} \tag{3-6}$$

The definition of the Hopfield network given in equation (3-2) is substituted in equation (3-6) yielding the time rate of change of the Hopfield network energy as

$$\dot{E}(t) = -\dot{V}^T \dot{U}. \tag{3-7}$$

Rewriting equation (3-7) in terms of the summation definition of equation (3-1) yields

$$\dot{E}(t) = -\sum_{i=1}^{N} g'(u_i)\dot{u}_i^2. \tag{3-8}$$

Since $g(\bullet)$ is monotonically increasing, its derivative, $g'(\bullet)$, must always be nonnegative. The second term in the summation, $\dot{u}_i^2$, is also always nonnegative. Since the derivative of $E(t)$ is always nonpositive and $E(t)$ is lower bounded, it must tend to a minimum.

## C. THE HOPFIELD NETWORK AS A PARAMETER ESTIMATOR

The energy or cost function of the Hopfield network has been shown to have a finite minimum. Thus for any given set of connection weights ***T*** and biases ***C*** the output of the Hopfield network ***V*** converges to a minimizing solution. This behavior is analogous to the recursive least squares algorithm whose cost function was defined in equation (2-19). The cost function $J(\theta)$ of equation (2-19) is expanded to

$$\begin{aligned} J(\theta) &= \int_0^t e^{-\alpha(t-\tau)} x(\tau)^2 d\tau + \\ &\theta^T \left[ \int_0^t e^{-\alpha(t-\tau)} \phi(\tau) \phi^T(\tau) d\tau \right] \theta - 2 \left[ \int_0^t e^{-\alpha(t-\tau)} x(\tau) \phi^T(\tau) d\tau \right] \theta. \end{aligned} \tag{3-9}$$

As the first term is not a function of $\theta$ it has no effect on the minimization of $\hat{\theta}$ and may be discarded. Thus an equivalent cost function for RLS estimation is:

$$J(\theta) = \theta^T \left[ \int_0^t e^{-\alpha(t-\tau)} \phi(\tau) \phi^T(\tau) d\tau \right] \theta - 2 \left[ \int_0^t e^{-\alpha(t-\tau)} x(\tau) \phi^T(\tau) d\tau \right] \theta. \tag{3-10}$$

A comparison between the Hopfield network energy function, equation (3-4), to the simplified RLS energy function, equation (3-10), enables certain equivalencies to be made. Setting the Hopfield network output ***V*** equal to $\theta$, the following equations must be true:

$$T = -\int_0^t e^{-\alpha(t-\tau)}\phi(\tau)\phi^T(\tau)d\tau$$
$$C = \int_0^t e^{-\alpha(t-\tau)}x(\tau)\phi(\tau)d\tau. \tag{3-11}$$

Thus if the Hopfield network weight matrix and bias vector are set according to equation (3-11), the Hopfield network output will converge to the optimal estimate for $\theta$ as would the RLS algorithm.

## D. THE HOPFIELD NETWORK FOR DIRECT ADAPTIVE CONTROL

As shown above, the Hopfield network will provide an optimal estimate of $\theta$ provided that $T$ and $C$ are formed as per equation (3-11). Equation (2-17) specifies the formation of $\phi(t)$ as a vector of $y(t)$, $N-1$ derivatives of $y(t)$, $u(t)$, $N-1$ derivatives of $u(t)$, and the scalar $p^*(s)q(s)y(t)$. However, these derivatives may not be directly available to the control parameter identifier. Furthermore, analog differentiation is not a reliable operation in a real world environment as differentiators are highly subject to noise [Ref. 9:p. 99]. Rather then using the direct adaptive control equation directly, both sides of equation (2-15) may be operated on by

$1/p^*(s)q(s)$ yielding

$$\frac{q(s)}{p^*(s)q(s)}u(t) = \frac{h(s)}{p^*(s)q(s)}y(t) + \frac{k(s)}{p^*(s)q(s)}u(t) + g_p y(t). \qquad \textbf{(3-12)}$$

Equation (3-12) shows that $y(t)$ and $u(t)$ may be filtered by $1/p^*(s)q(s)$ and the resultant control parameters $h(s)$, $k(s)$, and $g_p$ remain the same. When $y(t)$ and $u(t)$ are properly operated on by a state space filter in controllable canonical form, the necessary derivative states are available without the need for a differentiator.

## E. CONVERGENCE AND STABILITY OF THE HOPFIELD NETWORK

As mentioned earlier, the Hopfield network based direct adaptive controller will converge to an optimal estimate of $\theta$. The particular transfer characteristic $g(\cdot)$ has no effect on the steady state value of $\hat{\theta}$ unless one or more of the actual elements of $\theta$ exceed the bounds of the nonlinearity. It is the responsibility of the designer to ensure that all controller parameters are within the bounds of the transfer characteristic.

This implementation of the Hopfield network attempts to operate the neurons in their linear region. Therefore, the terminology Hopfield network rather than neural network is more appropriate to this implementation.

### 1. The effect of T and C on Hopfield network convergence

Assuming that the Hopfield network output $\hat{\theta}$ does not exceed the bounds of the nonlinearity, then the Hopfield network output states behave as a linear system response to a step input where $T$ is analogous to the continuous time $A$ matrix and $C$ is analogous to $B$. The steady state value of $\hat{\theta}$ is $-T^{-1}C$ and the speed of the convergence is proportional to the eigenvalues of $T$. In order to speed convergence it is desirable to keep the eigenvalues of $T$ as large as possible.

There are several methods to speed the convergence of the Hopfield network. The first is to be aware of the operating conditions of the plant in its likely area of operation. The rate of convergence of the algorithm is determined by the eigenvalues of $T$, and therefore by the magnitude of $y(t)$ and $u(t)$. If these signals are small enough to affect the rate of convergence then they can be properly scaled to increase the eigenvalues of $T$. The next method is to increase both $T$ and $C$ by a scalar positive constant $\lambda$. This does not change the steady state value of $\hat{\theta}$ but it does increase the eigenvalues of $T$. In a digital simulation the use of $\lambda$ is almost unrestricted, however in an analog implementation its value is limited by the voltage and current capacities of the components and the noise level.

### 2. The Excitation of the Input Signal

The nature of the input signal *v(t)* is critical to parameter convergence. The signal *v(t)* must provide persistency of excitation in order to guarantee convergence of the parameters. Persistency of excitation is related to the frequency content of the signal [Ref. 5:p. 423]. For example, a sinusoid would not be persistently exciting for a system of order greater than two in the sense that it does not contain enough information to estimate the parameters. When an input signal is not persistently exciting, the eigenvalues of the ***T*** matrix tend to become small, slowing the Hopfield network convergence. Based on frequency content alone, the best input for persistent excitation is a white noise signal. However, white noise signals are not well suited for systems with small bandwidth. Since white noise is typically zero mean, a system with small bandwidth filters the white noise to a very small zero mean signal yielding a ***T*** matrix with very small eigenvalues. A superior input signal for use in these systems is a square wave of frequency suitable to the system since the square wave concentrates its energy in a finite bandwidth, whereas a white noise signal has an evenly spread power spectrum. The period of the square wave should be determined to ensure that it is suited to both the reference model and the plant.

## F. DIGITAL SIMULATION OF A HOPFIELD NETWORK

### 1. The Processing Element

The processing element is most easily represented in software as a single-input single-output system. The input is the sum of the product of the weights and the inputs to the neuron added to the bias term. The output of the neuron is simply the output passed through the transfer characteristic. This work will consider four different transfer characteristics: 1) the sigmoid, 2) the hyperbolic tangent, 3) the identity (a linear neuron), and 4) a saturation nonlinearity. It should be noted that all four of these transfer characteristics are monotonically increasing. The subroutine that applies the nonlinearity to the neuron input is called SIGMOID.M and is included in Appendix A.

### 2. The Hopfield Network

Having implemented the neuron, the implementation of the Hopfield network is a matter of implementing the nonlinear differential equation that describes a Hopfield network given in equation (3-2). The function that iterates a Hopfield network over one sampling period is described below; the corresponding MATLAB function is included in Appendix A as HOPFIELD.M. If the neuron is linear (case 3 above), then $\boldsymbol{U}=\boldsymbol{V}$ and equation (3-2) becomes a simple state space linear differential equation and may be simulated in a single time step by converting the continuous model to a discrete model and iterating the discrete model by one time step. The value returned is the Hopfield network output at the next time step.

The nonlinear neurons are not as simple. The MATLAB routine ODE45 was written to solve a nonlinear differential equation written in state space form [Ref 10:pp. 3-137 to 3-139]. This routine was modified to operate directly on the Hopfield network nonlinear differential equation (3-2). The only problem with this method is its computational complexity with respect to the linear method. The sigmoidal and hyperbolic tangent Hopfield networks were simulated in this manner.

The saturation nonlinearity Hopfield network was implemented similarly to the linear problem. The Hopfield network was determined for one iteration as described for the linear case. Following that solution, the output was passed through a saturation nonlinearity. Although this was not strictly the solution to the nonlinear differential equation, it was a very close approximation. If the sampling time was sufficiently small, the solution arrived at through this simplified method was indistinguishable from that arrived at via the more complex ODE45 solution and there was more than a fifty fold savings in simulation time.

# IV. THE HOPFIELD NETWORK FOR ADAPTIVE CONTROL

## A. THE ALGORITHM

The implementation of the Hopfield network for direct adaptive control is reasonably straightforward. A schematic of the Hopfield network controller is shown in Figure 4-1.

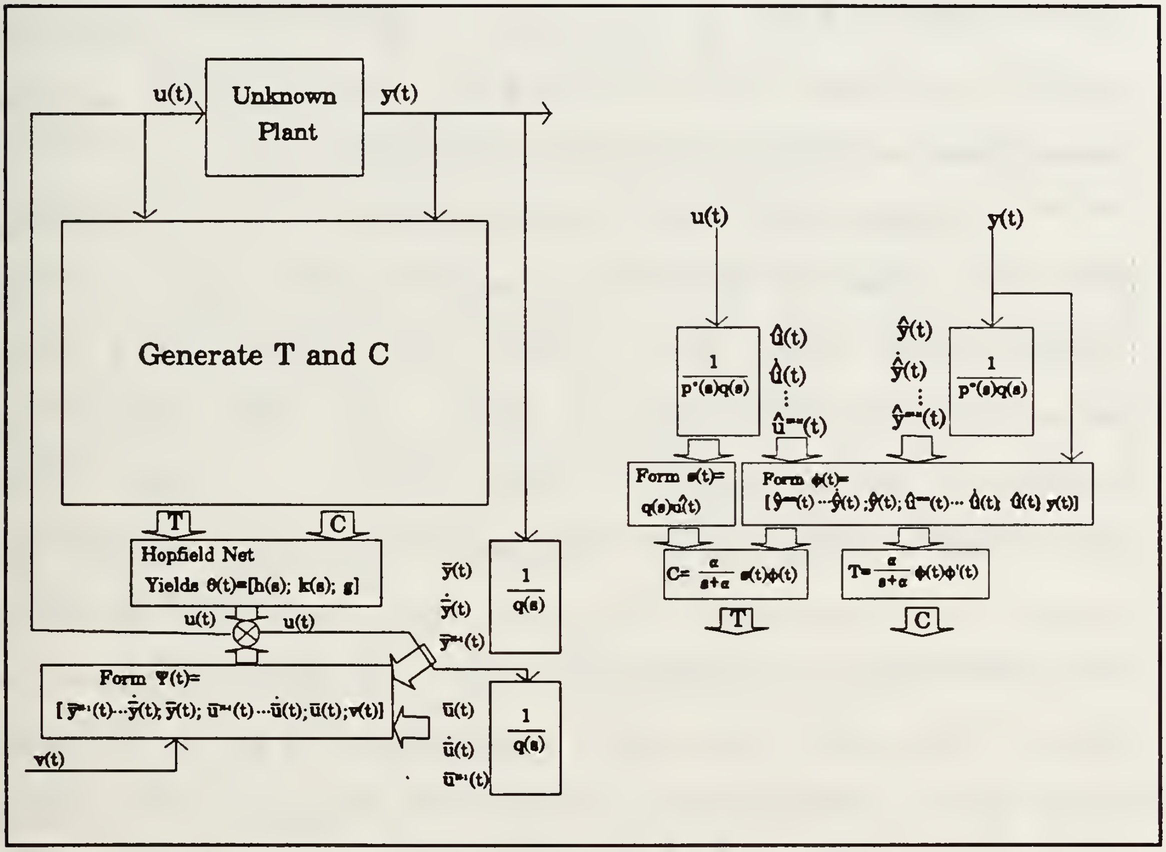

Figure 4-1 - Hopfield Net Adaptive Controller

The first step towards the eventual implementation of the Hopfield network controller in hardware was a high level software simulation. This simulation was intended to give insight into the behavior of a system controlled by a Hopfield network as well as to discover any potential difficulties inherent in this control implementation. The high level simulations were written in MATLAB. The next few sections describe the general steps in the implementation of the Hopfield network for direct adaptive control.

### 1. Determination of the System Order

An estimate of the system order *N* and the number of system zeros *M* must be determined. There are a variety of methods available to the designer, from complex systems identification tools to a simple educated guess.

An important note in the determination of the system order is a caution about the modeled system zeros. If the optimal plant model with *N* poles and *M* zeros at a given operating point has unstable zeros, then the closed loop system will not be stable at that point. In a linear system this implies instability, whereas a nonlinear system may transit to an operating region with stable zeros and oscillate around the unstable operating point. One possible method to control this plant is to use a plant model (*N* and *M*) of lower order than the one determined above. For example, if a third order linear plant with one unstable zero ($N=3$, $M=1$) were controlled by this direct adaptive control algorithm, it would be unstable. However, if the direct adaptive controller were designed with the assumption that the plant was third order with no zeros ($N=3$, $M=0$), in some cases the closed loop system would be stable because the unstable zero would never be canceled. Of course, this reduced form cannot model the unknown system as well as the full model thus reducing the accuracy with which the controller can follow the reference model. The accuracy with which a plant can follow the reference model is largely determined by how closely the reduced order optimal model matches the actual system.

### 2. Determination of the Reference Model and the Observer

The choice of the reference model, $p^*(s)$, is clearly critical to the closed loop system performance. Since it is desired that the closed loop system behave as the reference model to the input signal, a reference model must be chosen that exhibits the desired response. For this work we chose as reference models the class of Butterworth filters. The bandwidth $\omega_0$ is chosen according to the desired speed of response. This reference model exhibits fast rise time with small overshoot.

The observer $q(s)$ was chosen in accordance with traditional control theory. A fast observer (one with very large poles) is not necessarily good because it will follow the noise as well as the signal [Ref. 5:p. 260]. As a rule of thumb, it is a good choice for the observer polynomial to have four times the bandwidth of the reference model. In this thesis the observer was chosen as a Butterworth polynomial with bandwidth $4\omega_0$.

### 3. Determination of the Weight Filter Pole

The weight filter $\alpha/(s+\alpha)$ is associated with the forgetting factor used in the calculation of $T$ and $C$. The value of $\alpha$ is an important factor in the speed of convergence of the Hopfield network. For linear systems, the Hopfield network converges faster when $\alpha$ is made smaller. However, experimentation has shown that if $\alpha$ is decreased much beyond the slowest root of the plant, the speed of convergence will remain about the same. Thus a good choice for $\alpha$ for the control of a linear or nearly linear plant is the magnitude of the slowest root of the plant.

The choice of $\alpha$ in a nonlinear plant is somewhat more difficult. For many nonlinear plants the above guideline for linear plants for choosing $\alpha$ is still valid. However, if the plant linearization changes more rapidly than the slowest root of the linearization, then $\alpha$ should be chosen to be somewhat larger. This increases the 'forgetting factor' of the Hopfield network's weights, allowing the network to concentrate on the more recent inputs to the system and forget about the older, less valid data.

### 4. Determination of the Input and Output Data Filters

The filter for the input $u(t)$ and output $y(t)$ data is determined as shown in equation (3-12). Two vector variables, $\phi_y(t)$ and $\phi_u(t)$, are formed by filtering $y(t)$ and $u(t)$ through a filter with dynamics *1/p.(s)q(s)* in controllable canonical form:

$$\phi_y(t) = \begin{bmatrix} s^{2N-M+1}y_f(t) \\ \vdots \\ sy_f(t) \\ y_f(t) \end{bmatrix}; \qquad \phi_u(t) = \begin{bmatrix} s^{2N-M+1}u_f(t) \\ \vdots \\ su_f(t) \\ u_f(t) \end{bmatrix} \tag{4-1}$$

where

$$y_f(t) = \frac{1}{p^{*}(s)q(s)}y(t); \qquad u_f(t) = \frac{1}{p^{*}(s)q(s)}u(t). \tag{4-2}$$

### 5. Determination of the Control Signal Filter

The control system output from the Hopfield network based direct adaptive controller is given in equation (2-13). Both sides may be divided by *q(s)* to yield the control signal

$$u(t) = \frac{h(s)}{q(s)}y(t) + \frac{k(s)}{q(s)}u(t) + g_p v(t) \tag{4-3}$$

where *h(s)*, *k(s)*, and $g_p$ are the components of the parameter vector $\hat{\theta}$. Therefore, *y(t)* and *u(t)* must be filtered by *1/q(s)* in order to generate the control signal. This is done as before, with two *1/q(s)* state space filters in controllable canonical form.

## B. A LINEAR SYSTEM

Before proceeding to nonlinear systems, some basic tests of the Hopfield network based direct adaptive controller were made on a linear system. The linear system chosen was a third order pitch and depth model of the AUV. The state space form of this system is

$$\begin{bmatrix} \dot{q} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -0.07 & -0.04 & 0 \\ 1 & 0 & 0 \\ 0 & -0.12 & 0 \end{bmatrix} \begin{bmatrix} q \\ y \\ z \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u(t) \qquad \textbf{(4-4)}$$

where $q$ is the pitch rate, $y$ the pitch, and $z$ the depth of the AUV [Ref. 1:pp 27-30]. It is desired to control the depth state to match a reference model's response to an external input $v(t)$. Figure 4-2 shows plots of the baseline run with the ***T*** and ***C*** filter pole $\alpha$ set to one radian per second, and $\lambda$ set to one. Figure 4-2a shows the reference model's and the actual system's response to the input signal. Figure 4-2b is an expanded view of the portion of Figure 4-2a outlined by the box. Clearly, this is not a satisfactory control as the output does not follow the reference model. The problem stems from the fact that the Hopfield network estimation of $g_p$ drops to nearly zero, and because $g_p$ is the coefficient that amplifies the input signal $v(t)$, this reduces the magnitude of the input to the plant. The reduction of input yields a ***T*** matrix with extremely small eigenvalues, slowing convergence of the Hopfield network to a glacial pace.

The solution to this problem is to prevent $g_p$ from falling below a certain threshold. The threshold value may be

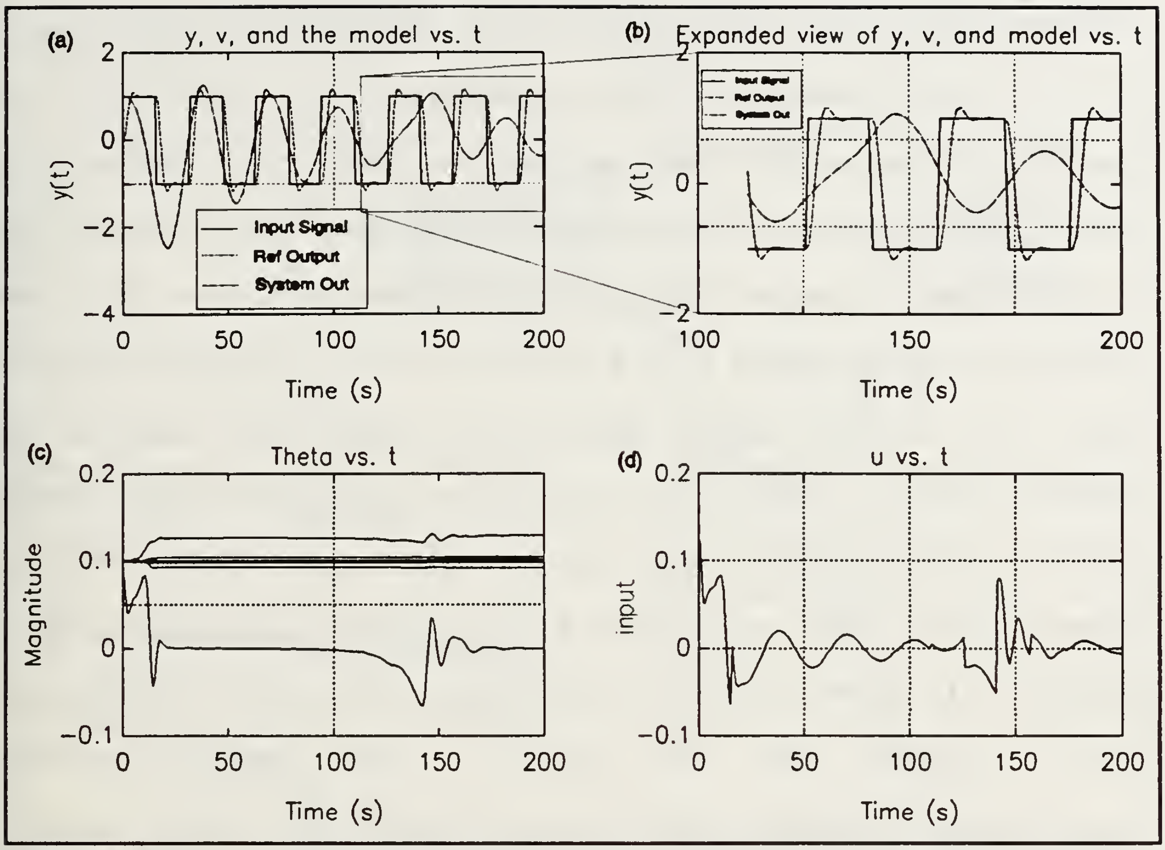

**Figure 4-2** - Hopfield control of a Linear System ($g_p$ not limited, $\lambda=1$)

determined by using system identification routines to estimate a model for the unknown system. Noting that $g_p$ is equal to $1/r_1$, where $r_1$ is the highest order numerator coefficient of the unknown system, the threshold value may be some fraction of the estimate of $g_p$ or $\beta/r_1$. The value of $\beta$ is set according to the confidence in the estimate of $r_1$. A $\beta$ of one implies absolute confidence that the actual $g_p$ is no lower than the estimate. A $\beta$ of between 0.1 and 0.5 is more realistic as it allows some room for error in the initial estimate of $g_p$. Nonlinear systems should have $\beta$ in this range because the estimate of $r_1$ may change over time. In practice the value of $\beta$ has little impact on the convergence as long as it is reasonably large but does not exclude the actual value of $r_1$ from consideration by the Hopfield network. For the next run of the linear system the Hopfield network estimate of $g_p$ was limited to $0.2\hat{g}_p$ where $\hat{g}_p$ was found with the routine FIND_HK.M. The rest of the system parameters remain the same as the previous run. Figure 4-3a shows the reference model output and the actual plant output due to the input signal $v(t)$ and Figure 4-3b shows an expanded view of the outlined area of Figure 4-3a to more clearly demonstrate the plant convergence. It is notable that just prior to the Hopfield network convergence, the AUV's depth state became very large and this is what caused the convergence to occur so abruptly. Figure 4-3c shows a plot of the parameter vector $\theta$ versus time. This plot shows that the parameter vector converges in approximately 45 seconds.

The final simulation that was run on the linear plant was to demonstrate the effect of increasing $\lambda$ to 1,000,000. The $g_p$ threshold was left in place as it is still required for rapid convergence. Figure 4-4a shows the reference model and the actual model and Figure 4-4b expands the outined area of Figure 4-4a and shows that the actual output does converge to the output of the reference model. It is again noted that the AUV dropped to a large depth value before the parameter vector

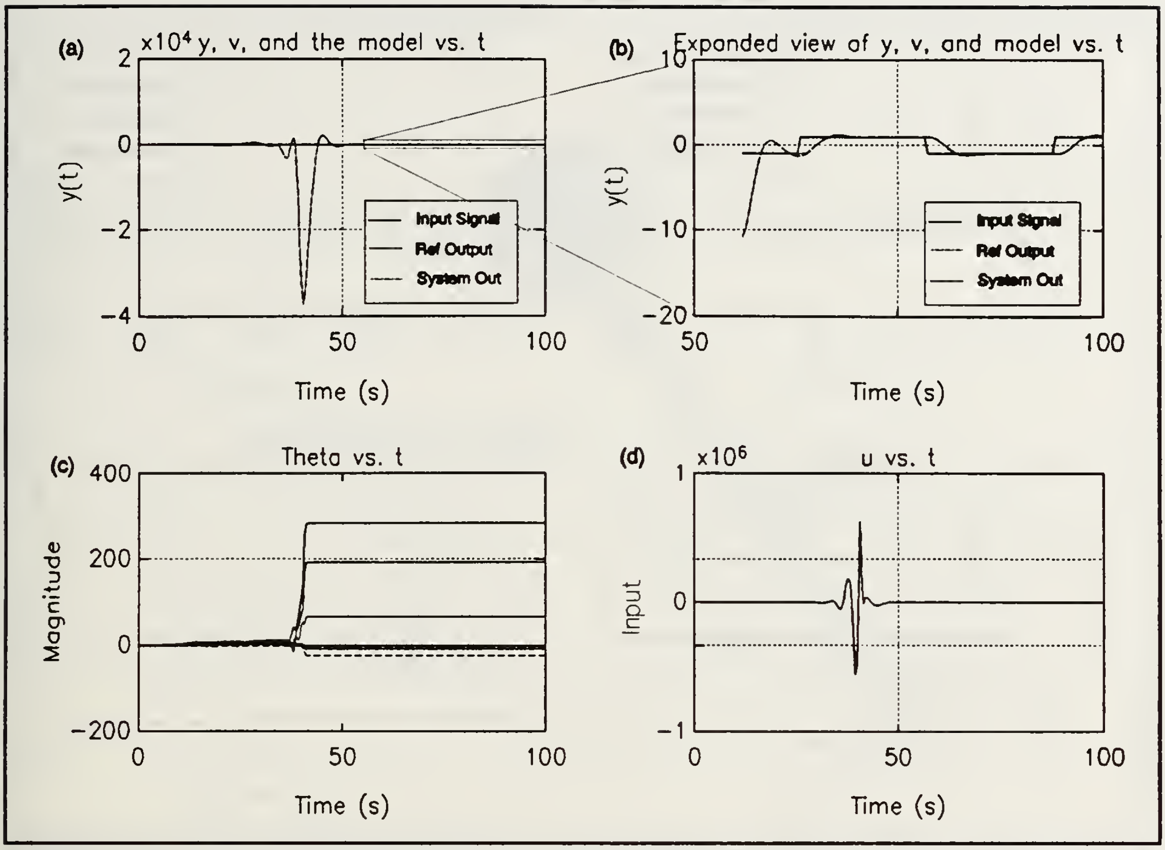

**Figure 4-3** - Hopfield Control of a Linear System ($g_p$ limited, $\lambda$=1)

converged. However, the depth excursion in this case is about 100 times less than in the previous simulation. The lower left plot is that of the parameter vector and shows that the parameter vector converges in about 10 seconds. This is a two fold improvement over the previous case where $\lambda$ was unity. This is as expected because the increase in $\lambda$ increased the eigenvalues of the $T$ matrix and thus the convergence of the Hopfield network. Although the use of $\lambda$ is practically unrestricted in a digital simulation, it is limited by the

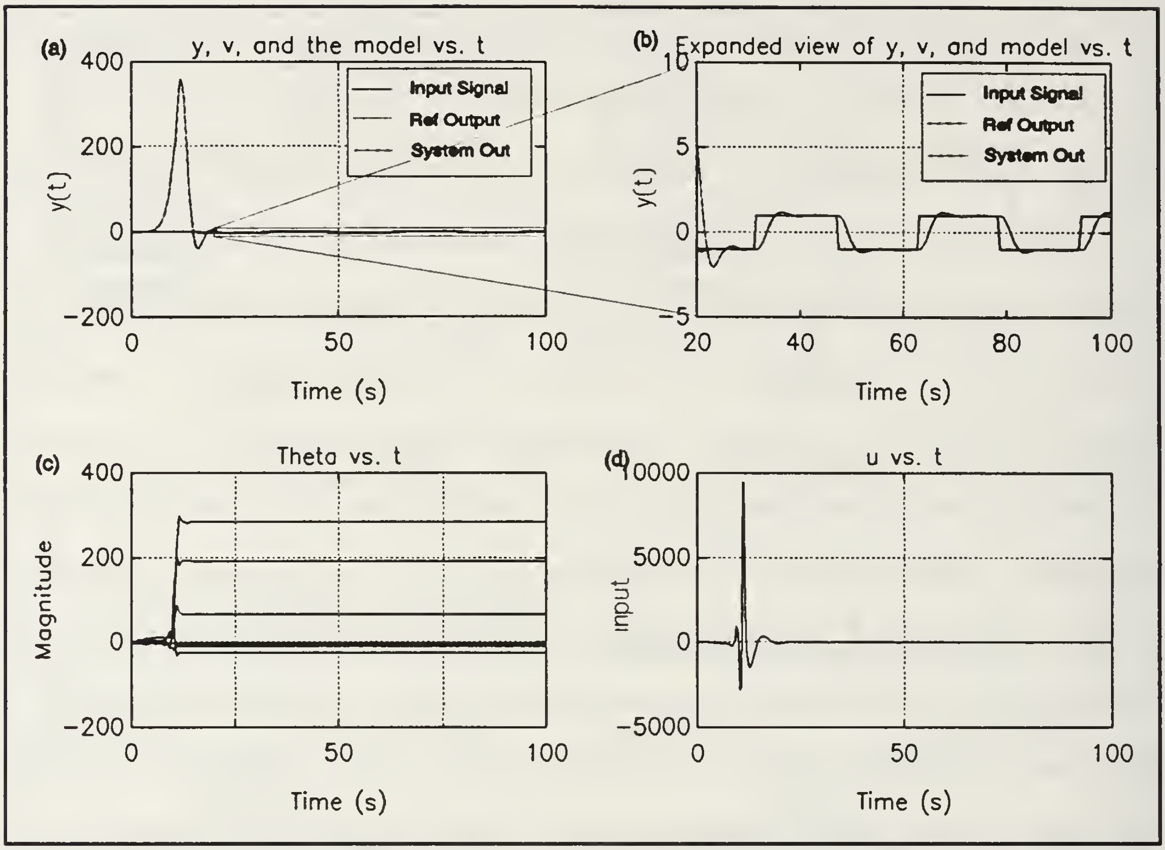

**Figure 4-4** - Hopfield Control of a Linear System ($g_p$ limited, $\lambda$=1,000,000)

available voltage and current magnitudes in an analog circuit implementation.

## C. THE INVERTED PENDULUM

The inverted pendulum was used as an initial test of the nonlinear direct adaptive Hopfield network controller. A simple model will be used with the pendulum swinging on a stationary axis and a control torque applied at the axis. Figure 4-5 shows this system. The nonlinear differential

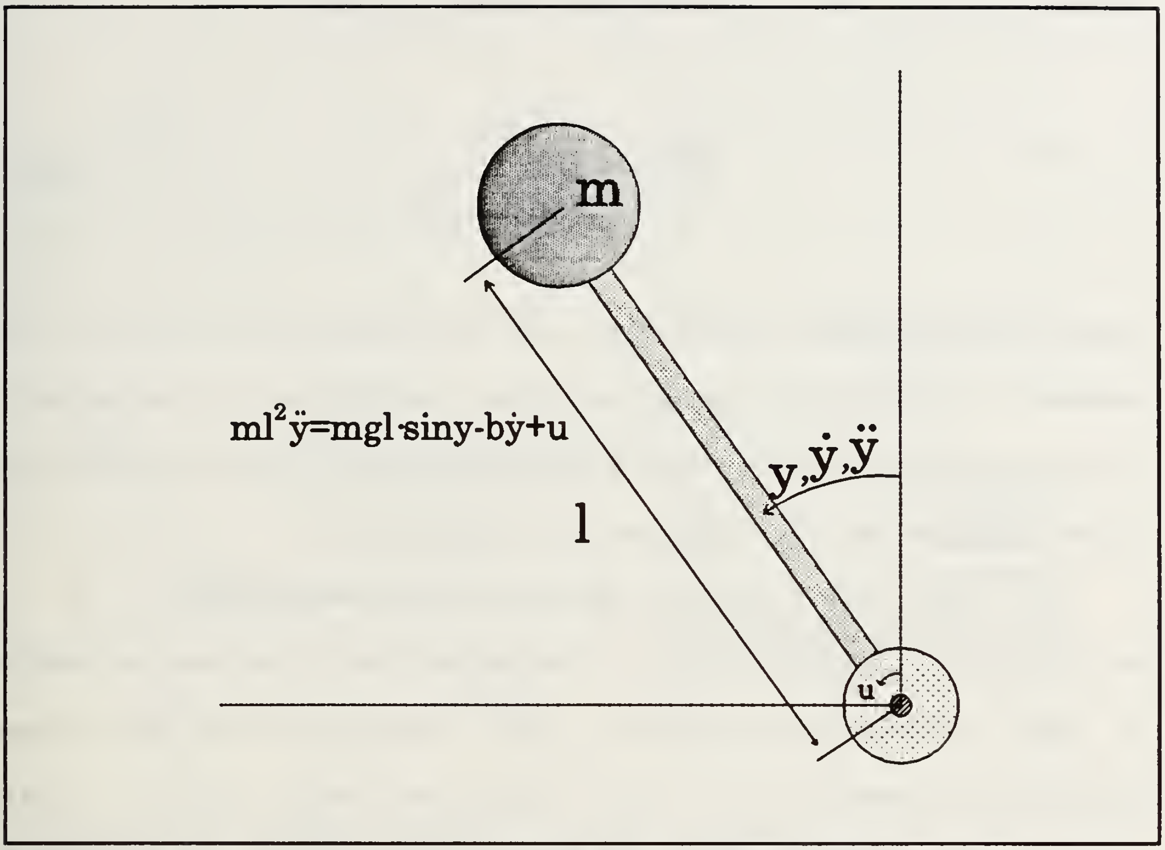

**Figure 4-5** - Diagram of an Inverted Pendulum

equation that describes this system is

$$\ddot{y}=\frac{g}{l}\cdot\sin y-\frac{b}{ml^2}\dot{y}+\frac{u}{ml^2} \qquad \textbf{(4-5)}$$

where *b* is the frictional coefficient, *m* is the mass of the pendulum and *l* is the length of the weightless arm. The goal of the control effort will be to maintain the position of the pendulum in a relatively upright position ($y\approx 0$). In this upright position the system of equation (4-5) may be linearized by letting *sin(y)=y* which yields

$$\frac{y(s)}{u(s)}=\frac{\frac{1}{ml^2}}{s^2+\frac{b}{ml^2}s-\frac{g}{l}}. \tag{4-6}$$

This linearization shows that the system order *N* is two, the number of zeros *M* is zero, and that the system is unstable at this operating point. The file PEND.M is the driver file for this problem and is included in Appendix A.

For this study the following values were used: $m$=.1 kg; $b$=1 kg-m$^2$/s; $l$=1 m. The elements of parameter vector $\theta$ were initialized to 0.1. The sampling time for these simulations was 0.1 s. The reference model was an $(N-M)^{th}$ order Butterworth filter of bandwidth 1 r/s. As discussed earlier, the observer was chosen as an $N^{th}$ order Butterworth filter of frequency 4 r/s. The pole of the filter for ***T*** and ***C*** was 1 r/s. The input signal was a square wave of magnitude 0.1 and frequency 0.1 r/s. The pendulum simulation was run with $\lambda=16\times10^{10}$. The large value for $\lambda$ was required to speed convergence to a reasonable amount of time. Figure 4-6a shows a plot of the pendulum angular position in radians, Figure 4-6b shows a plot of the control input *u(t)*, Figure 4-6c shows the time history of the Hopfield network output $\hat{\theta}(t)$, and Figure 4-6d shows the plot of the reference signal *v(t)*, the reference model response to *v(t)*, and the actual output of the

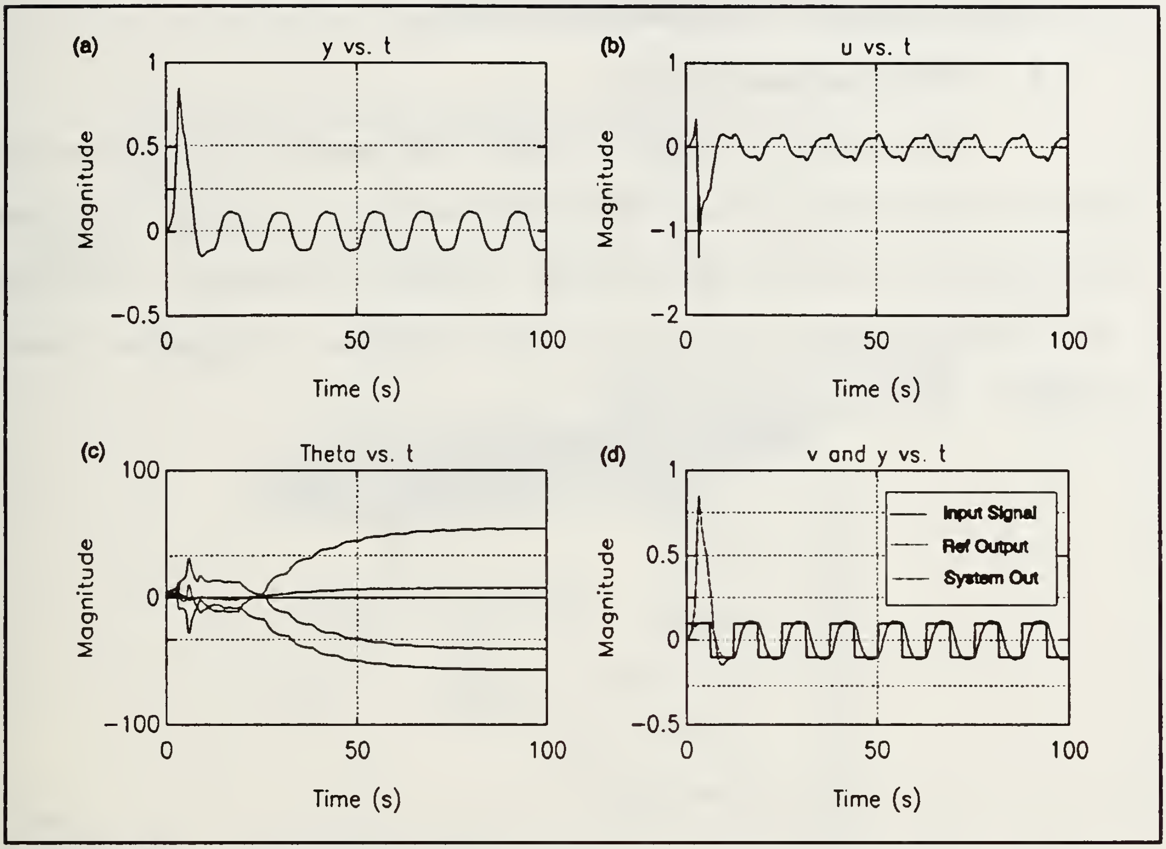

**Figure 4-6** - Baseline Convergence of the Inverted Pendulum

system. The convergence of $\hat{\theta}(t)$ is clearly evident in this figure. The last plot also shows that the system output converges to the reference model output as desired.

Several other Hopfield network control simulations were attempted on the inverted pendulum with mixed results. In cases where the pendulum fell upside down, the Hopfield network had difficulty in restoring the pendulum to an upright position. The problem lies in the fact that this is a model based implementation and that the Hopfield network must be able to adjust its weights as quickly as the system changes its linearization.

## D. THE AUTONOMOUS UNDERWATER VEHICLE

### 1. AUV Fundamentals

The model of the AUV that was used in this study is that of the seven foot NPS AUV [Ref 11] written in C and compiled for use in MATLAB [Ref. 12:pp. 124-129]. The model is nonlinear with six degrees of freedom, 12 states, and 5 inputs. The 12 states are:

$$\begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \\ x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} \Rightarrow \begin{matrix} \textit{surge (longitudinal speed)} \\ \textit{sway} \\ \textit{heave} \\ \textit{roll rate} \\ \textit{pitch rate} \\ \textit{yaw rate} \\ X \\ Y \\ Z \textit{ (depth)} \\ \textit{roll} \\ \textit{pitch} \\ \textit{yaw (course)} \end{matrix} \tag{4-7}$$

The five inputs are: the stern and bow rudder angles, the stern and bow planes, and the shaft RPM. A diagram of the AUV is shown in Figure 4-7.

The high level control effort envisioned for the AUV is to control the posture of the vehicle (the posture is made up of the course, the x and y position, and the depth) to

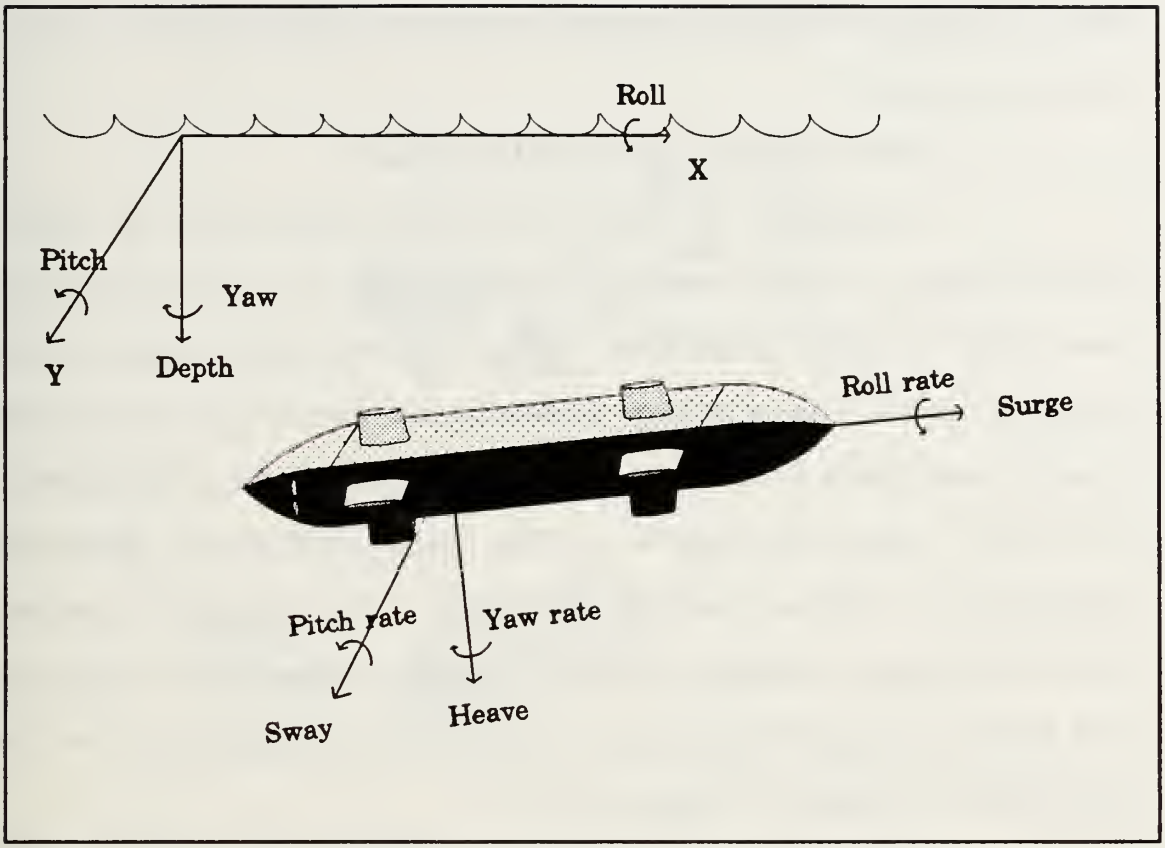

**Figure 4-7** - The NPS AUV

follow a reference posture. For simplicity, the stern and bow actuators were treated as one control input, with the bow actuator receiving the negative of the signal sent to the stern actuator.

### 2. A Control Scheme for the AUV

Because the AUV is a multiple-input multiple-output nonlinear system, the control scheme is complex. The upper layer of control is the path-following controller which receives as input the posture of a reference point and the AUV, and outputs AUV state reference signals for use by the low level controller. The low level controller receives the state reference signals and determines the appropriate control force to apply.

### a. The Path Following Algorithm

Although the path following algorithm is not a major issue in this work, it is included as a possible high level AUV control algorithm. The control scheme implemented in this model was a three dimensional extension of the two dimensional path following algorithm described by Kanayama et al [Ref 13:pp. 384-389]. The path following algorithm compares the AUV's posture with that of a reference posture and determines suitable state reference signals to maintain the AUV on the reference path. A pictorial description of the postures is shown in Figure 4-8.

The AUV is controlled by ordering the course rate, depth rate, and forward speed determined by the path following algorithm from the reference posture and the posture error. The path following algorithm produces three command signals based on the error posture between the actual vehicle and a reference model:

$$\begin{bmatrix} u_{command} \\ r_{command} \\ \dot{z}_{command} \end{bmatrix} = \begin{bmatrix} u_{ref}\cos\psi_e + K_x x_e \\ r_{ref} + u_{ref}(K_y y_e + K_\psi \sin\psi_e) \\ \dot{z}_{ref} + K_z z_e \end{bmatrix} \tag{4-8}$$

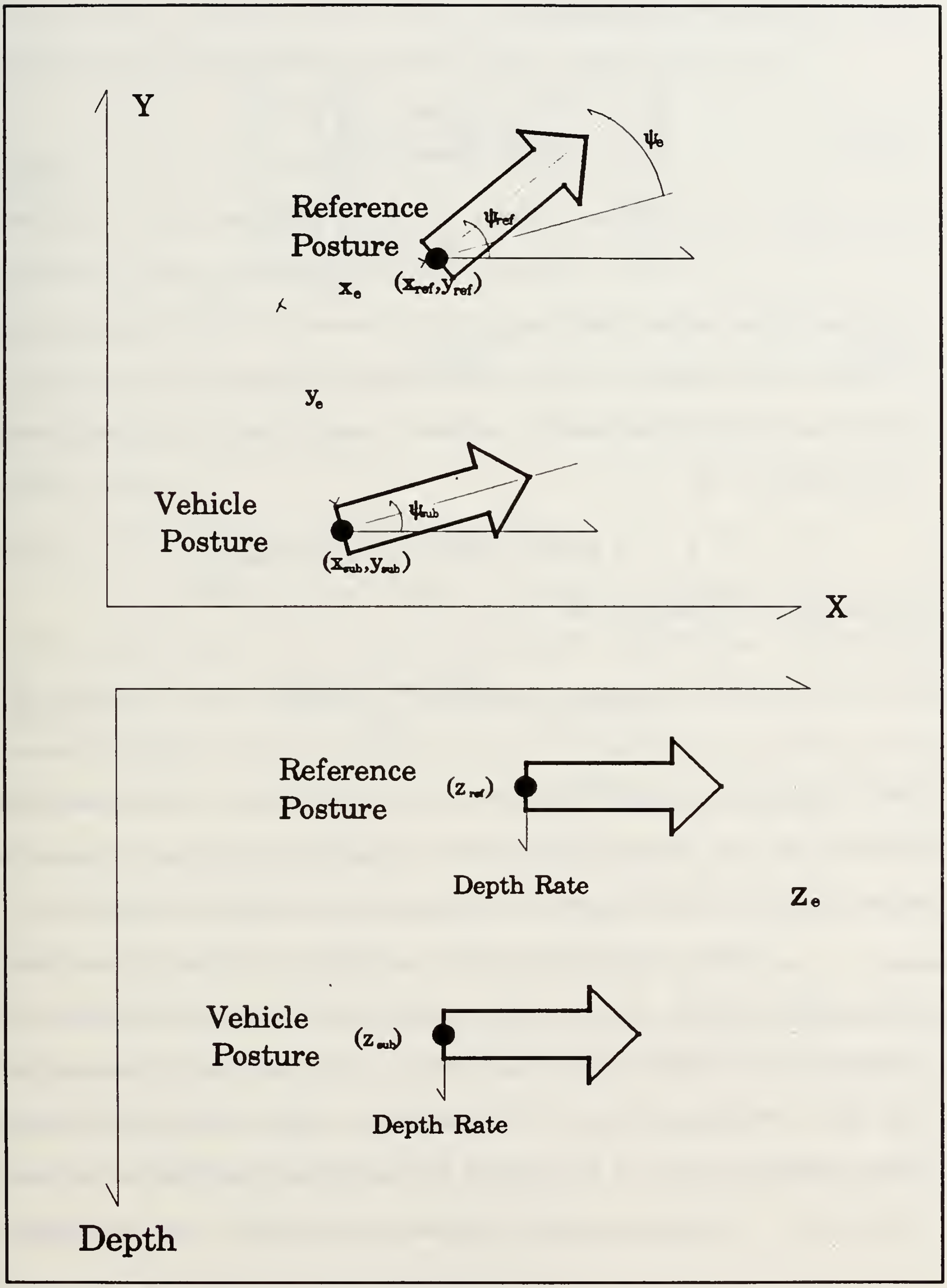

Figure 4-8 - Posture Definitions

where the error posture is

$$\begin{bmatrix} x_e \\ y_e \\ \psi_e \\ z_e \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi & 0 & 0 \\ -\sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{ref}-x \\ y_{ref}-y \\ \psi_{ref}-\psi \\ z_{ref}-z \end{bmatrix}. \qquad \textbf{(4-9)}$$

It should be noted that the third command is depth change rate and not the heave of the vehicle and therefore it is not a state of the AUV model. This signal may be calculated analytically as

$$\dot{z} = -u\ \sin\theta + w\ \cos\theta\sin\phi + v\ \cos\theta\cos\phi \qquad \textbf{(4-10)}$$

or readily estimated as

$$\dot{z}(kT_s) \approx \frac{z(kT_s)-z(kT_s-T_s)}{T_s} \qquad \textbf{(4-11)}$$

where $T_s$ is the sampling interval. The goal of the control system is to follow the three command signals given in equation (4-8) by properly adjusting the control inputs.

When the heading error is much less than one radian the coefficients $K_x$, $K_y$, $K_\psi$, and $K_z$ may be interpreted and determined in the following manner. $1/K_x$ is the time constant for the correction of the position error along the AUV's longitudinal axis. $K_y$ and $K_\psi$ are related coefficients that determine a second order transfer function of the error resolution across the AUV's longitudinal axis. The differential equation for the cross range error is

$$\ddot{y}+K_{\psi}v_{ref}\dot{y}+K_{y}v_{ref}^{2}y=0. \qquad \textbf{(4-12)}$$

Assuming the desired transfer function is to be critically damped, then $K_y$ and $K_\psi$ are related as

$$K_{\psi}=2\sqrt{K_{y}}\,. \qquad \textbf{(4-13)}$$

Lastly, $1/K_z$ is the time constant for the correction of the depth error.

### *b. A Linear Model of the AUV*

There are three system outputs that are controlled by the path-following algorithm described previously. The surge of the vehicle is controlled by the RPM command input while the depth rate is controlled by the stern and bow planes and the yaw rate by bow and stern rudders. The system order *N*, the number of zeros *M*, and an estimate of the gain $r_1$ must be known for each of the three sub-systems before proceeding with the Hopfield network control system. These values were estimated by generating an input and corresponding output sequence for each of the sub-systems and using a system identification routine to estimate a linear model for the sub-system. The routines used to determine the system model were GET_MOD.M and FIND_MOD.M included in Appendix A.

### *(1) The Course Rate Controller*

An input signal for the rudders was generated as a square wave of magnitude 0.4 radians and frequency 0.075 r/s with the vehicle travelling at 2 ft/s. The input signal was applied to the AUV model and the course rate output was observed. Figure 4-9a shows a plot of the input/output data used to determine a linear model of the rudder to course rate transfer function. Figures 4-9b and c present comparisons of the two best models. A summary of the results of the system identification are shown in Table 4-1. Table 4-1 shows that the mean absolute state error for the first order system

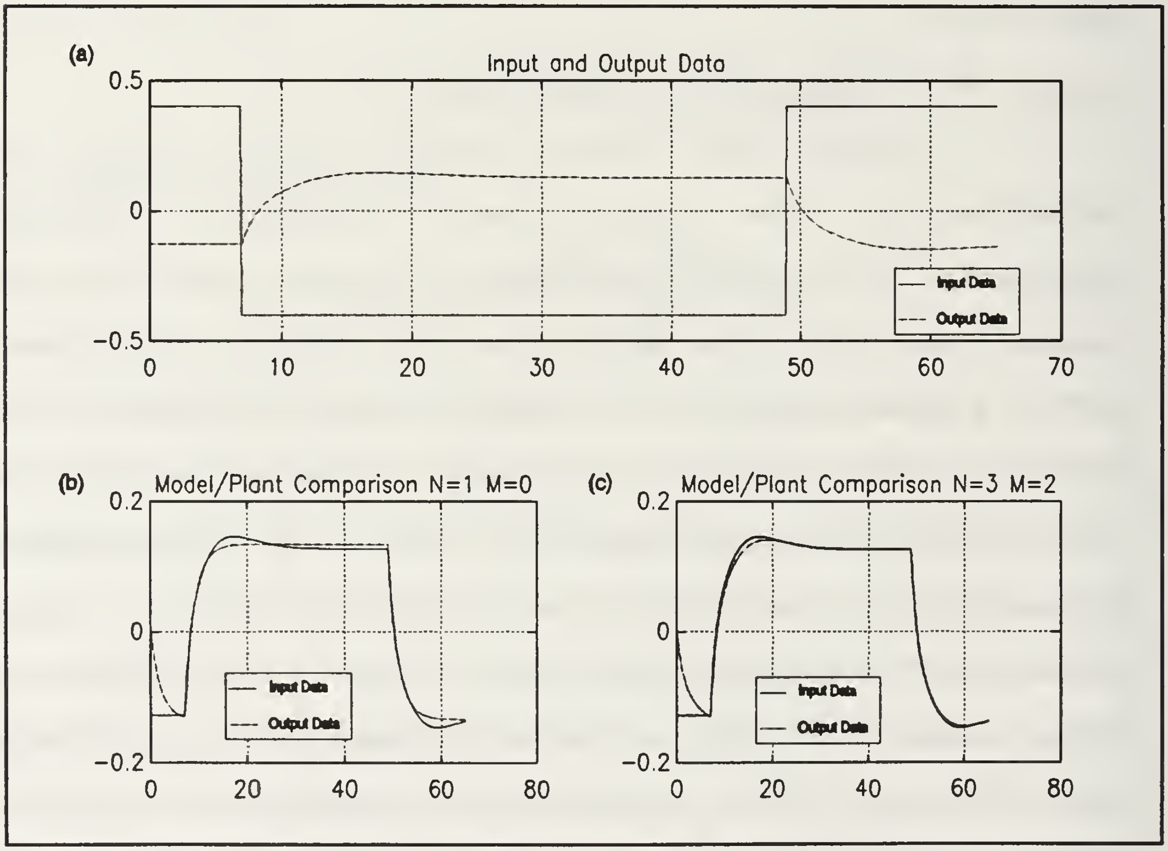

**Figure 4-9 - Course Rate Data and Resultant Models**

is the second lowest. A third order system with two zeros had the smallest error. Figures 4-9b and c show a comparison between the first order model and the third order model. Although the third order model is more accurate, the first order model is satisfactory. This reduction of model order significantly reduces the size of the computational problem because the size of the *T* matrix and thus the size of the Hopfield network varies as $(2N+1)^2$. The transfer functions of the first and third order models of the AUV's course rate state were estimated as

| N | M | Mean Abs Error |
|---|---|---|
| 1 | 0 | 0.0095 |
| 2 | 0 | ∞ |
| 2 | 1 | 0.0148 |
| 3 | 0 | ∞ |
| 3 | 1 | ∞ |
| 3 | 2 | 0.0060 |

**Table 4-1 - System Identification of Course Rate Data**

$$\frac{y_{course}(s)}{u_{rudder}(s)} = \frac{-0.1672}{s+0.499}, \qquad \textbf{(4-14)}$$

and

$$\frac{y_{course}(s)}{u_{rudder}(s)} = \frac{-0.1875s^2-0.1021s-0.01724}{s^3+1.313s^2+0.3859s+0.05419}. \qquad \textbf{(4-15)}$$

*(2) The Depth Rate Controller*

A square wave of amplitude 0.4 radians and frequency 0.025 r/s was applied to the dive planes with the vehicle travelling at 2 ft/s. The resultant depth rate as well as the input signal are shown in Figure 4-10a.

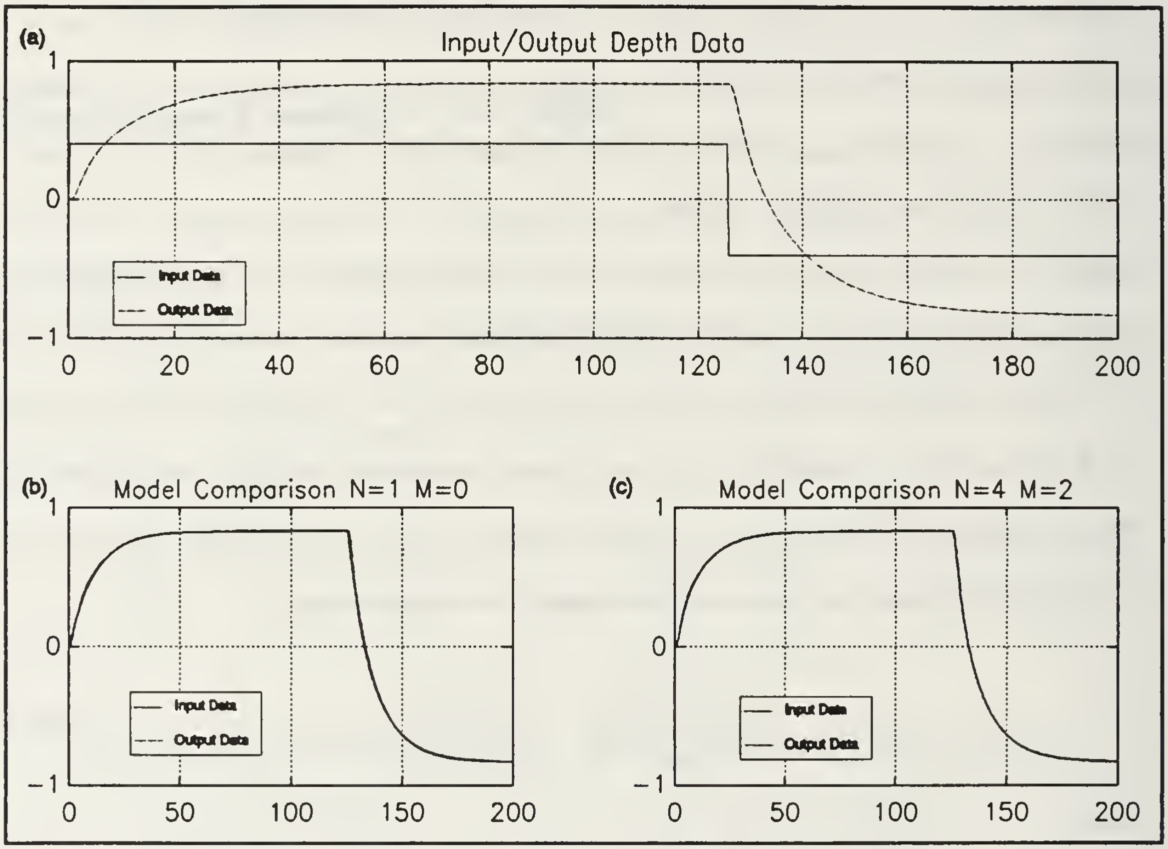

**Figure 4-10 - Depth Rate Data and Resultant Models**

The output data suggests that the system is first order. The input and output data used with the system identification routine GET_MOD.M and the results are shown in Table 4-2. A graphic comparison of the first and fourth order model is shown in Figures 4-10b and c. Again the first order model has the second smallest summed absolute state error. A fourth

| N | M | Mean Abs Error |
|---|---|---|
| 1 | 0 | 0.000341 |
| 2 | 0 | 0.000895 |
| 2 | 1 | 0.000895 |
| 3 | 0 | 0.000687 |
| 3 | 1 | 0.000661 |
| 3 | 2 | 0.000661 |
| 4 | 0 | ∞ |
| 4 | 1 | 0.00101 |
| 4 | 2 | 0.000120 |
| 4 | 3 | 0.000120 |
| 5 | 0 | 0.000873 |

**Table 4-2** - System Identification of Depth Rate Data

order model with two zeros had the smallest error. Although the fourth order model is slightly more accurate in terms of mean absolute error, the first order model performance is almost indistinguishable from that of the fourth order model. The transfer functions of these models for the AUV's depth rate were estimated as

$$\frac{y_{\dot{depth}}(s)}{u_{planes}(s)} = \frac{0.1832}{s+0.08809}, \qquad \textbf{(4-16)}$$

and

$$\frac{y_{\dot{depth}}(s)}{u_{planes}(s)} = \frac{0.1006s^2+1.607s+0.4324}{s^4+6.522s^3+8.238s^2+3.131s+0.2079}. \qquad \textbf{(4-17)}$$

*(3) The Speed Controller*

The input signal for the speed model was a square wave with a maximum magnitude of 480 RPM, a minimum magnitude of 260 RPM, and a frequency of 0.05 r/s. The AUV model was driven with this signal and the resultant input and output signals are shown in the Figures 4-11a and b respectively.

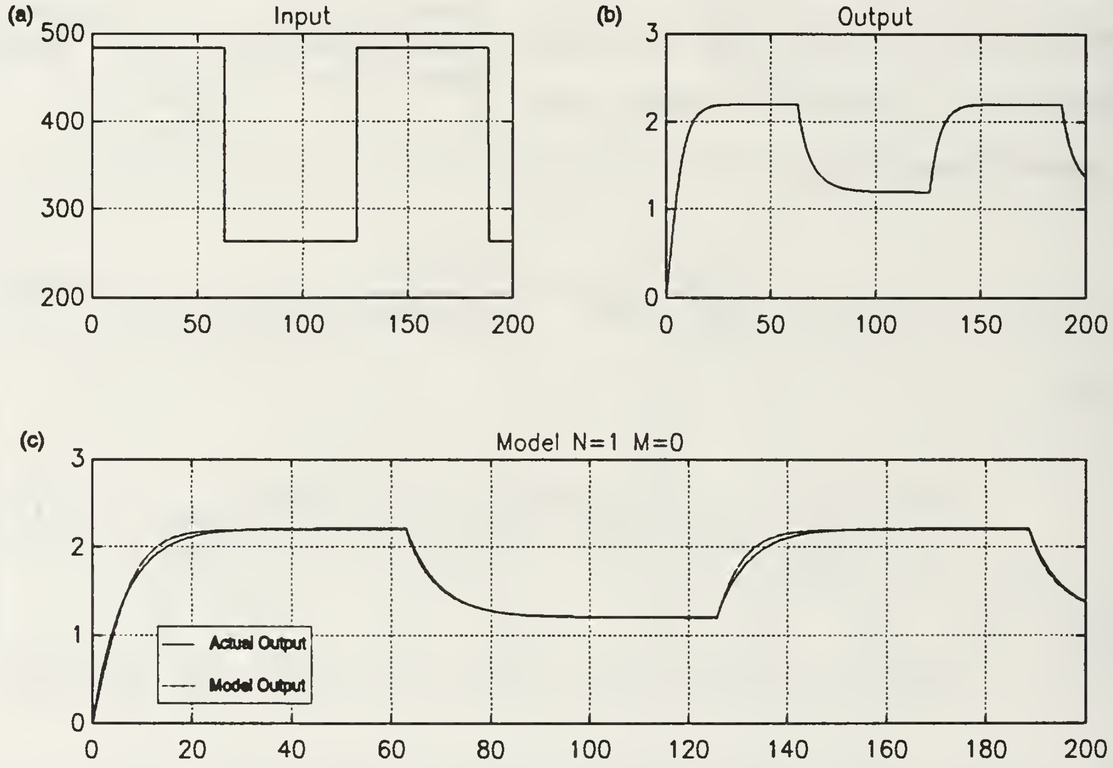

**Figure 4-11** - Speed Data and Resultant Models

The system identification routine GET_MOD.M was again used to generate a set of linear models for the above input and output data sequences. The results of the system identification are shown in Table 4-3.

Since the first order model had the smallest mean absolute state error of the models considered, no higher order models were used for comparison. The estimated transfer function of the first order model of the AUV's surge state is

| N | M | Mean Abs Error |
|---|---|---|
| 1 | 0 | 0.00110 |
| 2 | 0 | 0.00147 |
| 2 | 1 | 0.00110 |
| 3 | 0 | 0.00445 |
| 3 | 1 | ∞ |
| 3 | 2 | 0.00128 |
| 4 | 0 | 0.00468 |
| 4 | 1 | ∞ |
| 4 | 2 | ∞ |

**Table 4-3 - System Identification of Speed Data**

$$\frac{y_{speed}(s)}{u_{RPM}(s)} = \frac{0.0007173}{s+0.1566}. \qquad \textbf{(4-18)}$$

A comparison of this model to the actual system are shown in Figure 4-11c.

#### c. *Summary of Control*

The path follower outputs command signals for the control of the AUV's course rate, depth rate, and surge. Three first order direct adaptive control Hopfield networks were used to provide a suitable control signal to the actuators. Each of the three states is modeled as a first order system (*N=1, M=0*) and the estimated models of these systems may be used with the routine FIND_HK.M to determine initial estimates of the control polynomial parameters output by the Hopfield network.

### 3. Limitations of the Control Scheme

The piecewise linearity of the AUV is highly dependent upon its forward speed [Ref. 1: pp. 25-62]. At very low forward speeds (i.e. at less than 1 ft/s) the control surfaces have little effect on the motion of the AUV making the system appear uncontrollable and rendering the Hopfield network controller and path follower ineffective. Provisions must be made to prevent attempted convergence of the Hopfield network under these poor conditions.

### 4. The AUV Control Simulation

For the control simulation, a reference point moves along a prescribed path in a three dimensional space. The goal of the path following control is to generate three reference signals to keep the AUV as close to the reference point as possible. These three reference signals were passed to the Hopfield network controller in order to maintain the course rate, depth rate, and forward speed at the level of the reference signal filtered by the appropriate reference model $1/p^*(s)$. Thus the Hopfield network based direct adaptive control of the AUV requires three reference models to be denoted $p^*_c(s)$ for course rate control, $p^*_d(s)$ for depth rate control, and $p^*_s(s)$ for the surge control. The determination of these reference models is clearly critical to system performance.

As in any real world application the available control force is limited. In the case of the AUV, the control force for course rate and depth rate control is limited by the physical geometry of the control surfaces. The control surface input is limited to about 23° or 0.4 radians of rotation. The shaft RPM input is limited to range between 110 RPM and 750 RPM. The lower limit on shaft RPM prevents the vehicle from slowing to speeds where the control surfaces lose effect. These control force limits restrict the choices available for the reference models. If a reference model is too fast, the control surface is unable to meet the model and the system oscillates around the ordered state. The designer must ensure that the reference models are reasonable for the system to be controlled. Initially, the three models were chosen as Butterworth polynomials with a cut-off frequency of 0.2 r/s. The three observer polynomials were chosen as Butterworth polynomials of frequency 0.8 r/s.

The path-following parameters are also important to ensure accurate reference point tracking. If the path-follower attempts to drive the physical system beyond its capability to respond, the system cannot properly follow the path. However, if the path follower does not drive the system hard enough, the path following is slow to react to errors. The initial choice of parameters was: $K_x=0.5$, $K_y=0.01$, $K_\psi=0.2$, and $K_z=0.5$.

The AUV was started at coordinates $X=0$, $Y=0$, and $Z=0$. The reference point was started at $X_{ref}=20$, $Y_{ref}=20$, and $Z_{ref}=20$ generating an initial position error of 28 ft and an initial depth error of 20 ft. The reference point course rate, depth rate, and forward speed were changed according to the schedule in Table 4-4:

| time (s) | course rate (r/s) | depth rate (ft/s) | forward speed (ft/s) |
|---|---|---|---|
| 0 | 0 | 0 | 2 |
| 75 | $\pi/20$ | 0.5 | 2 |
| 85 | 0 | 0.5 | 2 |
| 95 | $\pi/20$ | 0 | 2 |
| 105 | 0 | -0.5 | 3 |
| 110 | 0 | 0 | 3 |
| 200 | $\pi/50$ | 0 | 2 |
| 250 | 0 | 0 | 2 |

**Table 4-4** - Reference Point Schedule

The simulation was run and the resultant motion of the AUV in the XY plane is shown in Figure 4-12a. Figure 4-12b shows the motion of the AUV in the depth plane. Figure 4-12c shows the time history of the range from the vehicle to the reference point. These plots show that the AUV follows the reference point fairly well throughout the circuit. The AUV corrects well for the initial range error of 28 ft. The first two 90° turns are made at turn rates faster than the AUV model can match since the AUV's turn rate with full rudder deflection at 2 ft/s is $\pi/25$ r/s. The AUV cannot keep up with these tight turns and the AUV's distance from the reference

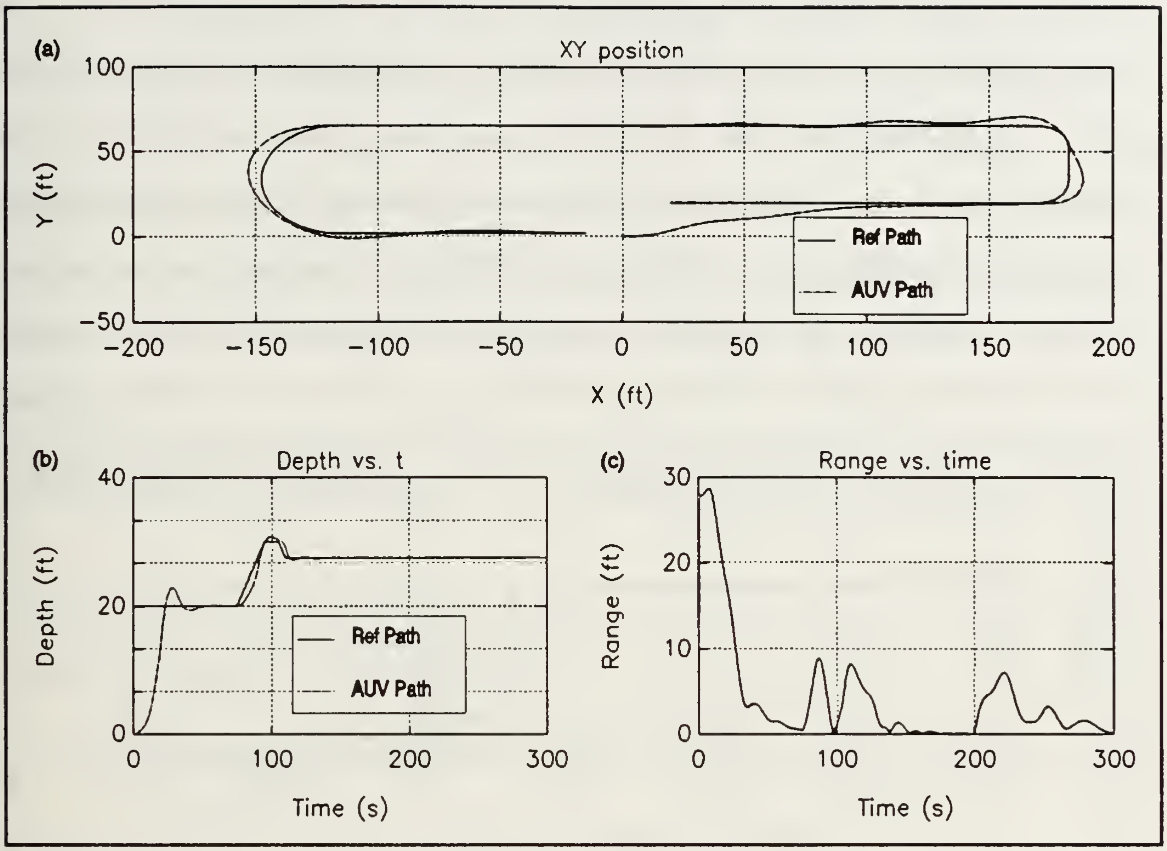

**Figure 4-12** - Initial AUV Path Following Simulation

point increases. After two tight turns the AUV resolves its position error during the long straight leg of the path. The last 180° turn is at a turn rate slow enough for the AUV to follow and yet there is still significant cross range error, inferring that the cross range error coefficients are not properly set. The plot of the AUV's depth reveals that the depth control is effective with small overshoot and reasonable settling time. The plot of the AUV's range from the reference position reveals that once the initial error has been resolved the AUV remains within eight feet of the reference point with the error increasing during reference point maneuvers.

Although the output of the Hopfield network was initialized to the predetermined coefficients of the *h(s)* and *k(s)* polynomials and the gain $g_p$, the linear models are not necessarily good representations of the AUV at varying speeds. Since the AUV is a nonlinear system, the Hopfield network output changes to adjust to new plant linearizations. Figure 4-13 shows plots of the parameter vectors for each of the three controls versus time. The plot of the parameter vectors shows the dynamic nature of the control parameter vectors. The control vectors are continually adjusted to best match the defined reference models.

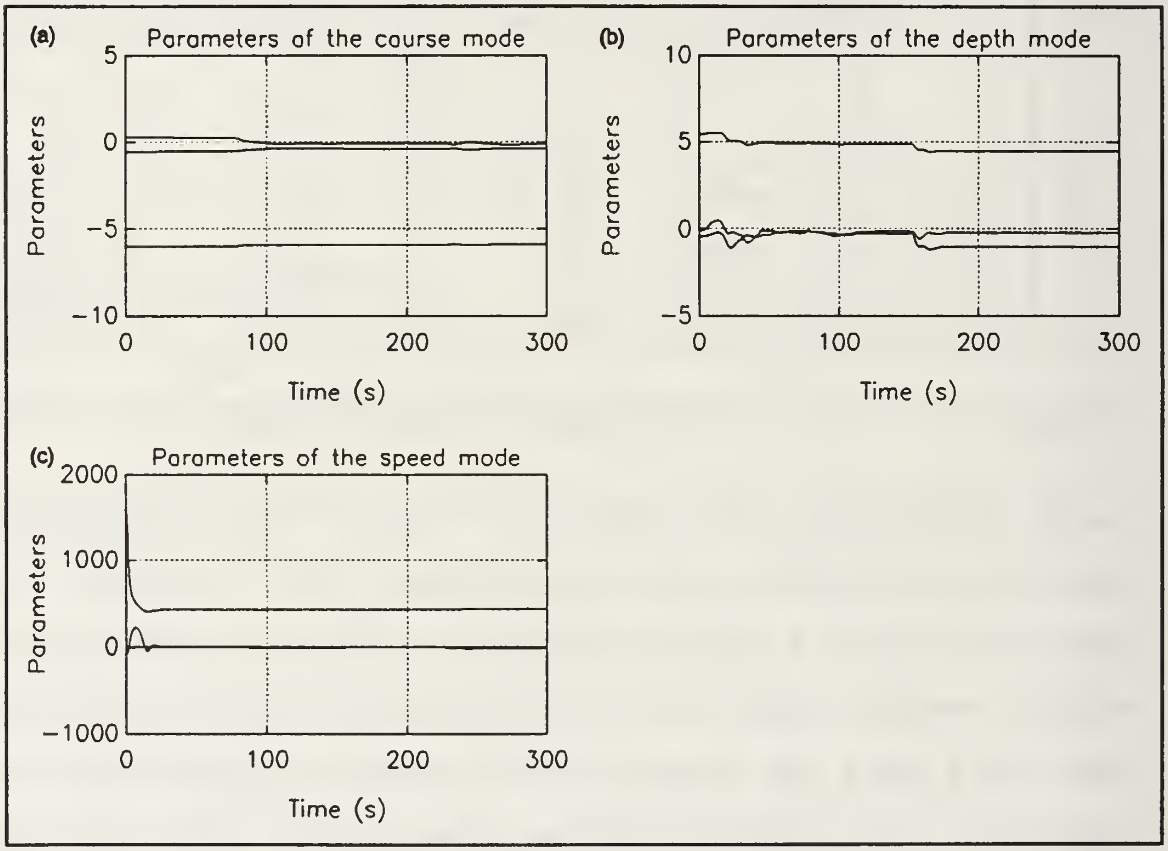

**Figure 4-13 - AUV Control Parameter Vector**

Figure 4-12 showed that the AUV did not satisfactorily follow a slow turn with the given path-following parameters. In order to improve the response the cross range error parameters were increased to $K_y=0.1$ and $K_{\psi}=0.63$. The path and initial conditions remain the same as the previous simulation. A plot of the system performance is shown in Figures 4-14a, b, and c.

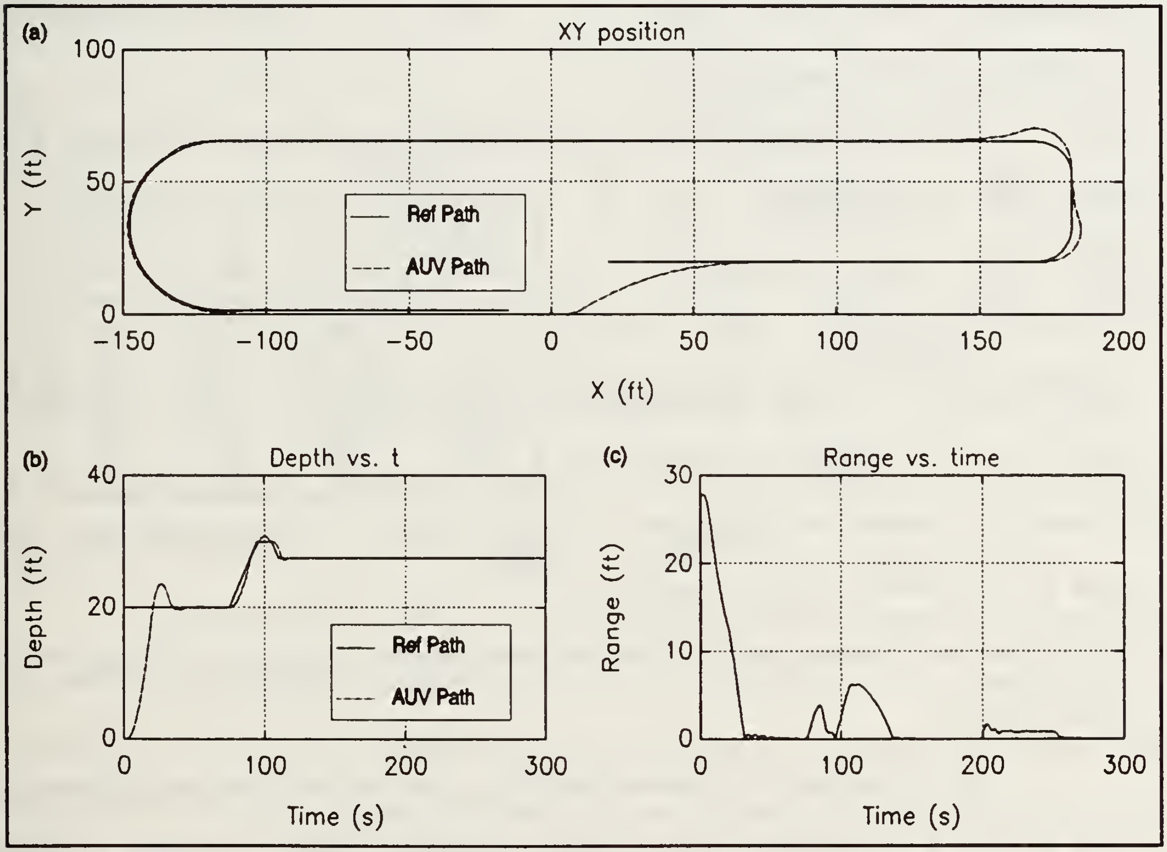

**Figure 4-14** - Improved AUV Path Following Simulation

The AUV's path in this simulation is clearly superior to that of the previous run. The AUV has completely overcome the initial position error within 30 seconds and maintains a much smaller range error during the wide turn.

The next group of simulations are run with a different form of path description. The preceding path was described by vehicle speeds, course rates, and depth rates. Although this method describes a path that is fairly easy for the vehicle to follow, it is relatively difficult for the path planner to input the path. A more typical form of path description is to define a set of waypoints and the time of arrival at each waypoint. This is a far easier method for a path planner to output a path description.

A set of waypoints were defined as shown in Table 4-5. The AUV simulation was run with the same constants and initializations as the previous run. The resultant path of the AUV in the XY plane is shown in Figure 4-15a and it shows that the AUV overshoots the waypoint and accumulates significant error at every turn.

| Time (s) | X (ft) | Y (ft) | Z (ft) |
|---|---|---|---|
| 0 | 20 | 20 | 20 |
| 75 | 160 | 20 | 25 |
| 105 | 160 | 70 | 20 |
| 200 | -60 | 70 | 20 |
| 250 | -60 | -80 | 20 |

**Table 4-5** - Waypoints for AUV Path

One method of reducing the overshoot is to reference the X and Y position error from a point that travels a fixed distance behind the reference point. This allows the AUV to 'look ahead' at the reference point and anticipate maneuvers. Several different following-ranges were tried and it was found that following six feet behind the reference point produced the best path based on mean range error. The system was

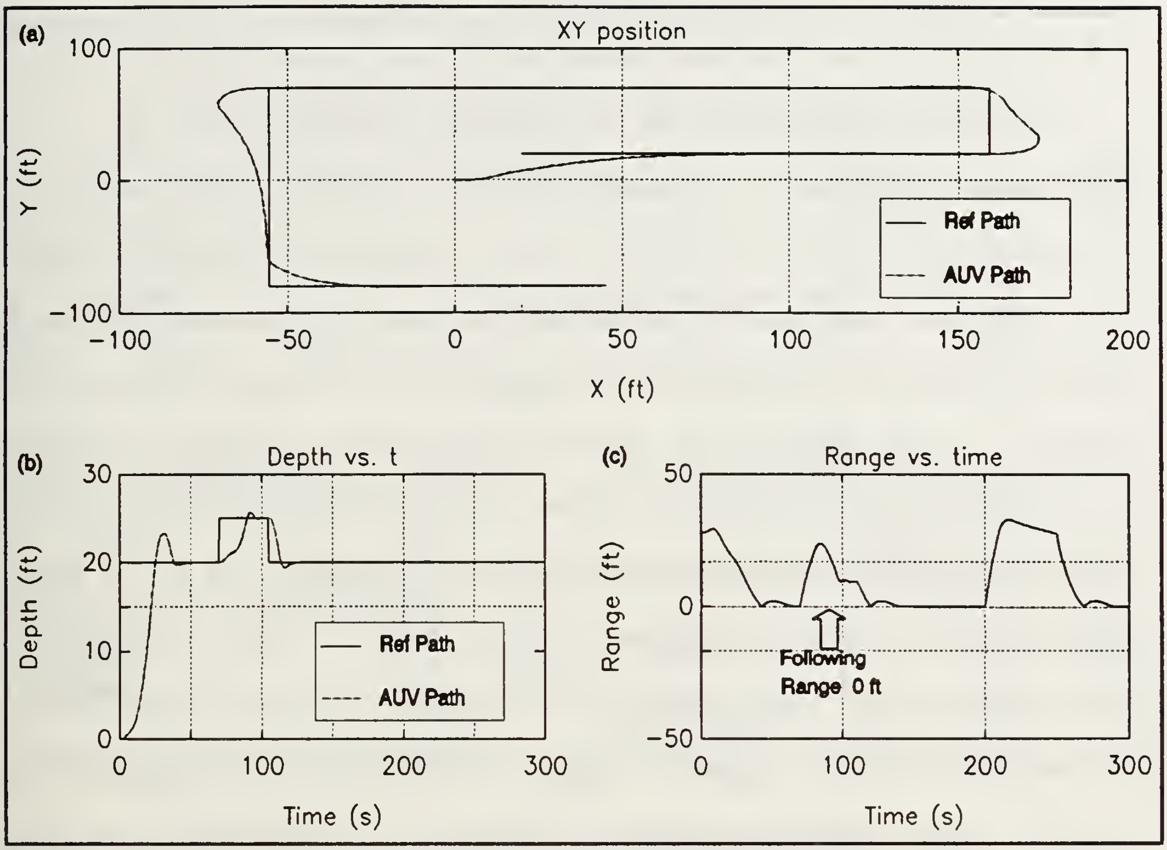

Figure 4-15 - AUV Traversal of Waypoint Path

simulated again with the waypoints from Table 4-5 and the resultant path in the XY plane is shown in Figure 4-16a.

The AUV follows this path more closely because the AUV is able to begin turning priore to reaching the waypoints. However, if it is necessary for the AUV to pass through the waypoints this range-following modification should not be used.

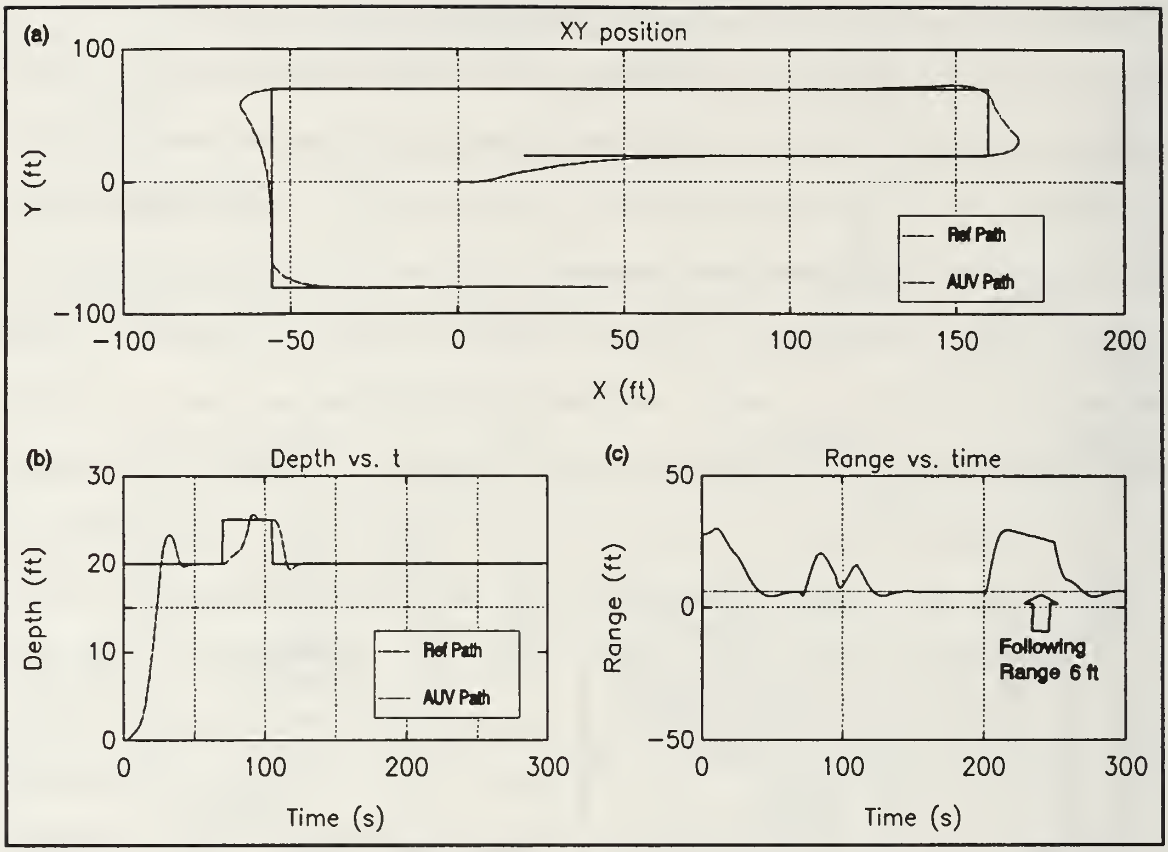

**Figure 4-16** - AUV Traversal of Waypoint Path, Following Distance Six Feet

# V. THE HOPFIELD NETWORK AS AN ELECTRONIC CIRCUIT

The major impetus for the use of a Hopfield network for direct adaptive control was the ability to implement the Hopfield network in analog hardware. This implementation was a straightforward translation of the Hopfield network based direct adaptive controller shown in Figure 4-1. The two fundamental components used were an operational amplifier (op amp) and a four-quadrant analog voltage multiplier. Since the controller was intended for use in a Very Large Scale Integrated (VLSI) circuit the op amps and multipliers were designed with complementary-symmetry metal-oxide semiconductor (CMOS) field-effect transistor technology [Ref 9:p. 773].

## A. A SIMPLIFIED FIRST ORDER SYSTEM CONTROLLER

Since all three AUV controls were modeled as first order systems this was the system order simulated. Before proceeding, we considered a simplified first order direct adaptive controller.

Given the uncertain first order system

$$\dot{y}(t) = -a\, y(t) + b\, u(t) \tag{5-1}$$

where $a$ and $b$ are unknown constants and the reference model

$$\dot{y}(t) = -p^{*}\, y(t) + v(t) \tag{5-2}$$

and equating them yields

$$-a\ y(t)+b\ u(t)=-p^{\bullet}\ y(t)+v(t). \tag{5-3}$$

The solution of *u(t)* from equation (5-3) is

$$u(t)=\frac{a}{b}y(t)+\frac{1}{b}(-p^{\bullet}\ y(t)+v(t)). \tag{5-4}$$

When *u(t)* is determined by equation (5-4), then the output *y(t)* tracks the output of the reference model given by equation (5-2). Since *a* and *b* are unknown they must be estimated.

The Hopfield network was used to estimate these parameters much as it was used to estimate the parameters for the higher order systems. Equation (5-1) was rearranged solving for *u(t)* and filtering both sides by an arbitrary first order stable monic polynomial *q(s)*

$$\frac{1}{q(s)}u(t)=\frac{a}{b}\,\frac{1}{q(s)}y(t)+\frac{1}{b}\,\frac{1}{q(s)}\dot{y}(t). \tag{5-5}$$

It was seen that the coefficients *a/b* and *1/b* present in equation (5-5) were the same as the unknown control coefficients in equation (5-4). Equation (5-5) is in the RLS estimation form of equation (2-15) where

$$x(t)=\frac{1}{q(s)}u(t); \qquad \phi(t)=\begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix}; \qquad \theta=\begin{bmatrix} \frac{a}{b} \\ \frac{1}{b} \end{bmatrix}. \tag{5-6}$$

Thus a two neuron Hopfield network may be used to estimate the parameters *a/b* and *1/b* necessary for the controller. A diagram of the first order Hopfield network controller is shown in Figure 5-1.

A simulation of this implementation was conducted using the simulation tool TUTSIM. The code for this simulation implementing the flow diagram of Figure 5-1 is included in Appendix B. The plant to be controlled was

$$y(t)=\frac{7.5}{s+5}u(t) \tag{5-7}$$

and the reference model was

$$y(t)=\frac{1}{s+1}v(t). \tag{5-8}$$

The pole of the observer *q(s)* was -2 r/s, the pole of the filter for *T* and *C* was 100 r/s, and both outputs of the Hopfield network were initialized to 0.1. The reference input *v(t)* was a unit magnitude square wave of frequency 0.05 r/s. The plot of the input and output of the system as well as the Hopfield network output are shown in Figure 5-2.

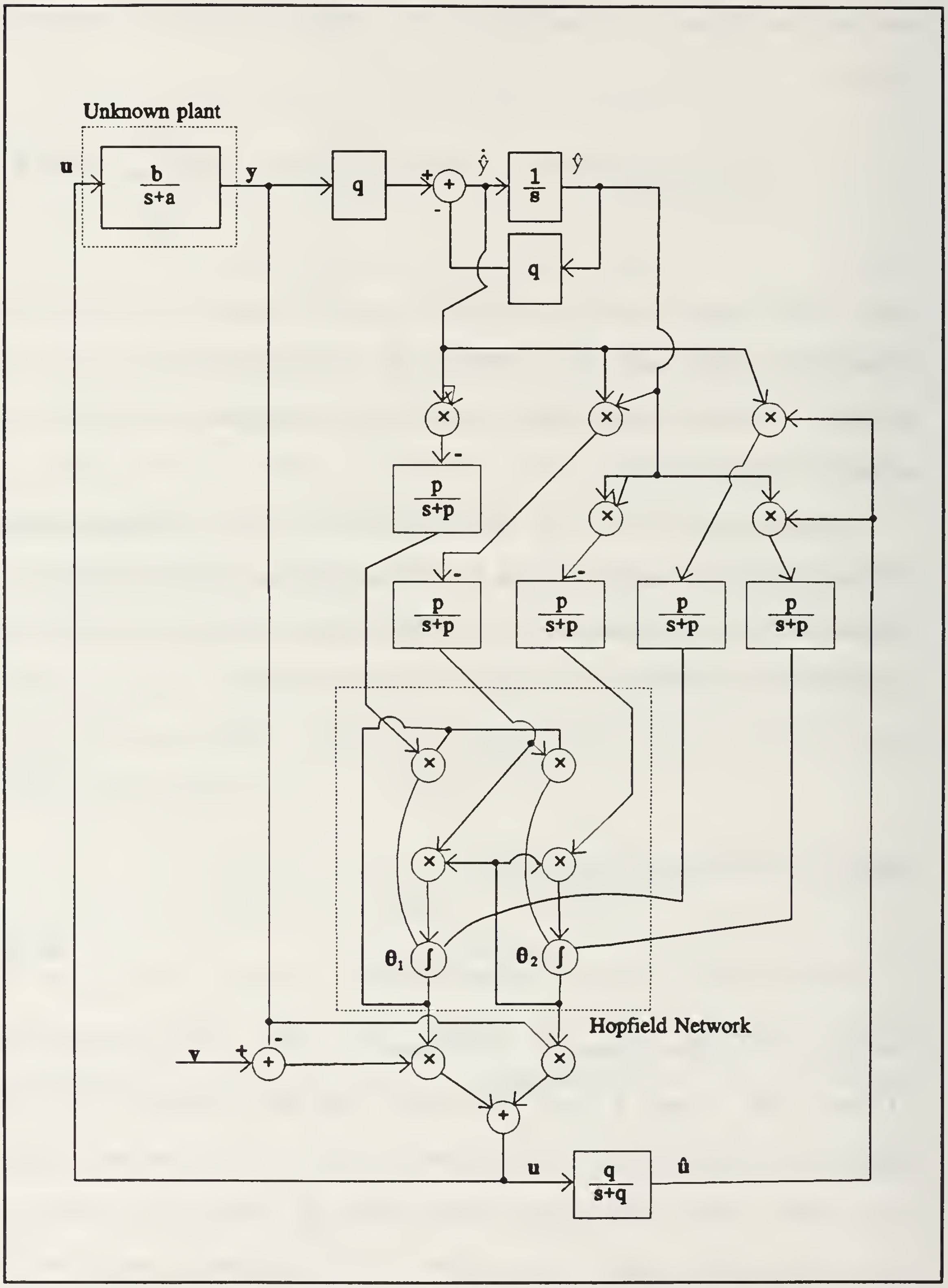

Figure 5-1 - First Order Direct Adaptive Hopfield Controller

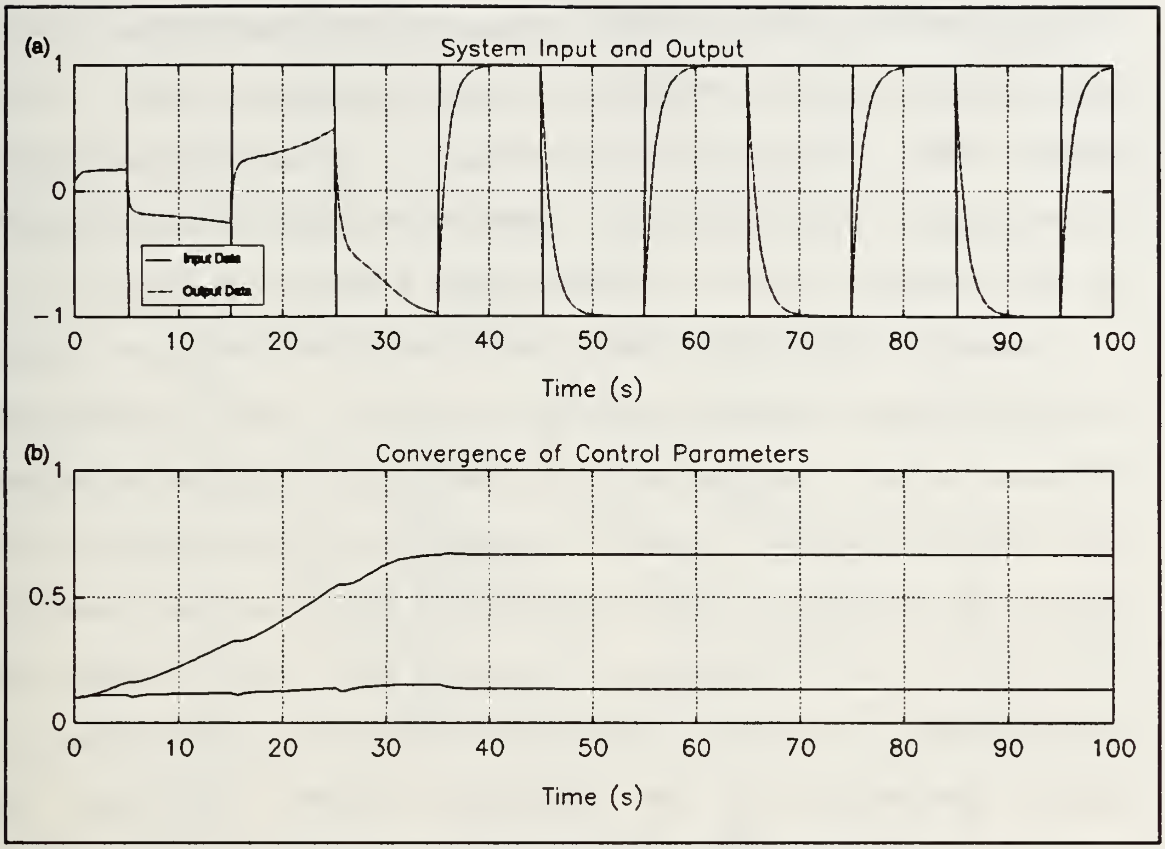

**Figure 5-2** - Results of First Order Hopfield Net

Figure 5-2a shows the input square wave $v(t)$ and the system output $y(t)$. The system output converges to the reference model output. Figure 5-2b shows the convergence of the parameters of the Hopfield network. The Hopfield network output converge to the vector $[0.13333\ 0.66667]^T$ which is the actual value of $\theta$.

## B. THE FIRST ORDER SYSTEM IN ANALOG HARDWARE

Figure 5-1 was converted to an analog circuit by replacing the integrators, summers, and gain blocks with the corresponding operational amplifier circuits [Ref. 14:pp. 35-

125]. The multiplier blocks were replaced by Analog Devices internally trimmed precision IC multipliers, model number AD534 [Ref. 15:pp. 6-27 to 6-35]. The actual Hopfield network portion of the circuit worked as expected, converging to the expected value of $\theta$ for a set value of $T$ and $C$.

However, when the closed loop system was run, the output of the Hopfield network tended to saturate. Once the Hopfield network output was saturated, the entire system saturated until the system was reset. A remedy for this problem is to scale the reference signal to maintain lower voltage levels in the system. Unfortunately, this would also tend to reduce the eigenvalues of the Hopfield network, slowing convergence. A SPICE simulation of the Hopfield network controller was designed to more closely analyze this problem.

## C. THE SPICE SIMULATION

The eventual goal of this work is to generate a single integrated circuit (IC) that holds the aforementioned circuitry. Since a CMOS design is best suited to analog VLSI circuits, all components were designed using CMOS technology.

### 1. The CMOS Op Amp

The CMOS op amp is a fairly common device. The example used in this work was chosen because of its simplicity as well as the availability of the SPICE parameters for the transistors [Ref 9:pp. 774-775]. General principles for the design of CMOS op amps are found in Reference 16.

### 2. The CMOS Four Quadrant Analog Voltage Multiplier

The design of the multiplier is based on the square law characteristic of the current-voltage curve of the CMOS transistor in saturation [Ref. 17:pp. 531-532]. Figure 5-3 is a diagram of the CMOS multiplier.

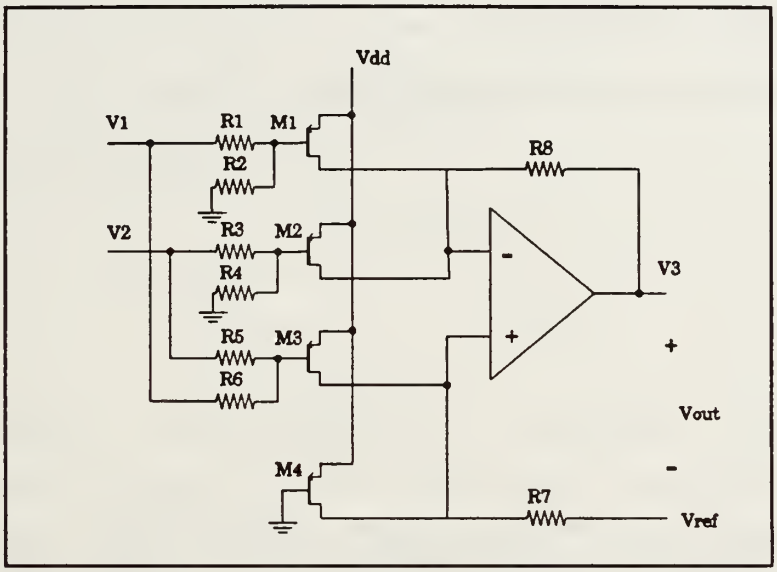

**Figure 5-3** - CMOS Analog Voltage Multiplier

The resistors $R_1$ through $R_6$ are identical and transform the input voltages $V_1$ and $V_2$ into the transistor input signals $V_1/2$, $V_2/2$, $(V_1+V_2)/2$ for input into transistor $M_1$, $M_2$, and $M_3$ respectively. Transistor $M_4$'s gate is grounded to provide a zero voltage reference signal.

The four CMOS transistors are p-channel devices that operate in saturation. The source currents of these transistors are

$$I_1=K\left(\frac{V_1}{2}-V^+-V_t\right)^2$$

$$I_2=K\left(\frac{V_2}{2}-V^+-V_t\right)^2 \qquad \textbf{(5-9)}$$

$$I_3=K\left(\frac{V_1+V_2}{2}-V^+-V_t\right)^2$$

$$I_4=K\left(-V^+-V_t\right)^2$$

and

$$K=\frac{WC_{ox}\mu_p}{2L} \qquad \textbf{(5-10)}$$

where $W$ and $L$ are the length and width of the transistor gate, $\mu_p$ is the mobility of holes, and $C_{ox}$ is the capacitance per unit area of the silicon dioxide gate. The output voltage referenced to $V_{ref}$ is

$$V_{out}=\frac{K}{2}V_1V_2R \qquad \textbf{(5-11)}$$

where $R$ is the resistance value of $R_7$ and $R_8$.

Thus the output of this device is a voltage difference proportional to the product of the two input voltages. A major problem with this implementation is that the voltage output requires a high gain differential amplifier on the output. Although this multiplier design is not completely satisfactory, it was used in the SPICE simulation to gain insight into the difficulties of building an analog circuit direct adaptive Hopfield network controller.

### 3. SPICE Simulation of the Hopfield Network

The first step in the SPICE implementation was the simulation of the linear two element Hopfield network noted in Figure 5-1. The inputs to the system were the ***T*** matrix and the ***C*** vector and were set to

$$\boldsymbol{T} = \begin{bmatrix} -1 & 0.1 \\ 0.1 & -2 \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}. \tag{5-12}$$

The theoretical solution of the Hopfield network was determined by solving for the steady state output of the step response of a state space system where ***A***=***T*** and ***B***=***C***. The steady state output was calculated as [Ref. 18:p. 688]

$$y_{ss} = -\boldsymbol{T}^{-1}\boldsymbol{C} = \begin{bmatrix} 1.0302 \\ 0.3015 \end{bmatrix}. \tag{5-13}$$

The SPICE simulation was run with $\hat{\theta}$ initially set to $[0.1 \; 0.1]^T$. A plot of the Hopfield network output is shown in Figure 5-4. The output of the Hopfield network converges to $[0.9968 \; 0.2865]^T$ which is within five percent of the theoretical values shown in equation (5-13) demonstrating that a Hopfield network is easily implemented with analog components.

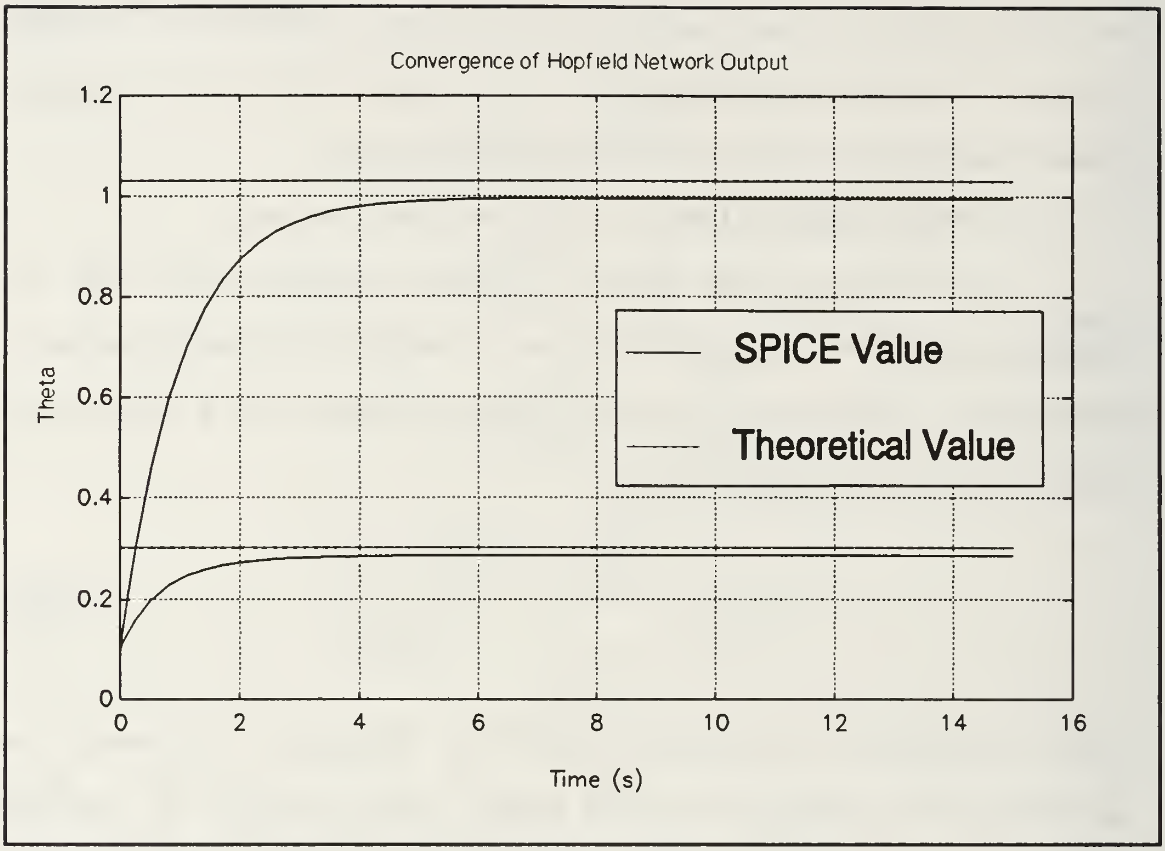

**Figure 5-4** - SPICE Hopfield Net Output

A SPICE circuit was designed to implement Figure 5-1 and is included in Appendix C. The major components of the circuit were written as subcircuits to improve the clarity of the code. The SPICE simulation showed that the multipliers representing the *T* and *C* matrices saturated causing the Hopfield network output to saturate as seen in the analog circuit mentioned earlier. One remedy to this problem is to scale the reference signal so that the internal signals do not saturate the analog devices at the cost of slowed convergence rate.

### D. REMARKS ON THE ANALOG CIRCUIT IMPLEMENTATION

The analog circuit implementation of this controller is theoretically possible, but the actual implementation in analog hardware proved more difficult. Since the multipliers and op amps saturate at relatively low levels, the system input signals may need to be scaled to ensure that all internal signals remain within saturation limits of the hardware. While the scaling process is straightforward, it tends to slow the rate of convergence of the Hopfield network.

There are other possible solutions to this problem that may not slow the convergence rate which need to be investigated before a reliable analog implementation of the Hopfield network based direct adaptive controller can be completed.

# VI. SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

## A. SUMMARY

The stability and convergence of a direct adaptive Hopfield network controller was shown for linear minimum phase systems. This result was extended to include nonlinear systems that could be modeled as piecewise linear minimum phase systems. Simulation studies of a linear system, an inverted pendulum, and the NPS AUV were included to examine the capabilities and limitations of the Hopfield network control scheme. Work on a suitable path following algorithm was included as a possible implementation of the direct adaptive Hopfield network control scheme.

The design of an electronic circuit to act as a Hopfield network was investigated. Computer simulations of a functional model of the circuit revealed no significant defects in the theory. However, an actual hardware model and a SPICE simulation of the controller did uncover several severe problems in the physical implementation of the circuit.

## B. CONCLUSIONS

The simulations of the direct adaptive Hopfield network controller revealed the following:

- In a digital simulation, the Hopfield network approach to direct adaptive control behaves similarly to the recursive least squares approach.

- The speed of convergence is highly dependent upon the magnitude and frequency content of the reference signal.

- The Hopfield network controller is a suitable controller for use with a complex multiple input multiple output nonlinear system. This controller also works well within the framework of a higher level controller such as the path following algorithm.

- The analog circuit implementation of the direct adaptive Hopfield network controller is feasible but is subject to the effects of the non-ideal analog components.

## C. RECOMMENDATIONS

This thesis laid the foundation for further work in the use of Hopfield networks for direct adaptive control. There remains a great deal of ground uncovered including:

- Improving the speed of convergence of the Hopfield network.

- The effect of disturbance and measurement noise on the convergence of the Hopfield network.

- Implementing the Hopfield network for control of non-minimum phase systems.

- Optimizing the AUV path follower parameters with respect to the Hopfield network controller reference models.

- Improving the analog circuit design of the direct adaptive Hopfield network controller to improve the circuit's resilience to non-ideal components.

- Generating an analog VLSI design for the Hopfield network.

- Writing code to provide for the automatic generation of a Hopfield network given the reference model, the observer polynomial, the system order, and the pole for the weight matrix filter.

# APPENDIX A. MATLAB SOFTWARE

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% FIND_HK.M %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [h,k,gp]=find_hk(a,b,pstar,q)
% FIND_HK.M - USAGE: [h,k,gp]=find_hk(a,b,pstar,q)
%       This function determines the direct adaptive control polynomials
%       h(s) and k(s) and the input gain gp given the system denominator
%       and numerator polynomials a(s) and b(s) respectively, the
%       reference model polynomial pstar(s), and the observer polynomial
%       q(s).
%
%       R. Scott Starsman 11-25-91

b=rlz(b);                                        % Remove leading zeros
bpstar=conv(b,pstar);
n=length(a)-1;                                   % System Order
m=length(b)-1;                                   % Number of Zeros

% Set up the Sylvester Matrix
Sa=[a';zeros(n-1,1)];
Sb=[zeros(n-m,1);b';zeros(n-1,1)];
for k=2:n
 Sa=[Sa [zeros(k-1,1);a';zeros(n-k,1)]];
 Sb=[Sb [zeros(n-m-1+k,1);b';zeros(n-k,1)]];
end
S=[Sa Sb];                                       % The Sylvester Matrix

f=conv(q, a-bpstar/b(1))';

% Determine solution to Diophantine Equation
hk=inv(S)*f(2:length(f));

% Determine controller polynomials and gain
h=hk(n+1:2*n);
k=hk(1:n);
gp=1/b(1);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% SIGMOID.M %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Y]=sigmoid(X,sigtype)
%SIGMOID  This function subjects the input vector X to the transfer
%         characteristic designated by sigtype.  The function is called
%         by: Y=sigmoid(X,sigtype).
%
%         sigtype specifies the transfer characteristic:
%               1 - Sigmoid
%               2 - tanH
%               3 - Linear
%               4 - Saturation
%
%         R. S. Starsman 5-1-91
%         Copyright (c) R. S. Starsman, 1991
%         All Rights Reserved

[n,m]=size(X);
if (sigtype==1)
  Y=ones(n,m)./(ones(n,m)+exp(-X));
elseif (sigtype==2)
  Y=(ones(n,m)-exp(-2*X))./(ones(n,m)+exp(-2*X));
elseif (sigtype==3)
  Y=X;
elseif (sigtype==4)
  Y=hard_lim(X,-1,1);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% HOPFIELD.M %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [V1] = hopfield(V0,T,C,Tf,sigtype,maxval,lambda)
%HOPFIELD  Iterates a Hopfield net for Tf seconds given the initial state
%       of the Hopfield net V0, the weight matrix T, the bias vector C,
%       the sigmoid type (sigtype), the scale factor, lambda (optional),
%       and the neuron saturation level, maxval (optional).
%
%       The matrix T must be negative definite.
%
%       sigtype specifies the transfer characteristic:
%               1 - Sigmoid
%               2 - tanH
%               3 - Linear
%               4 - Hard limited saturation
%
%      maxval is an optional parameter that specifies the upper and lower
%         bound on bounded neural outputs (sigtype=1, 2, or 4). The
%         default value is 1.
%
%       lambda is a scale function that multiplies the T and C matrices.
%         The larger lambda is, the faster the convergence.  The default
%         value is 1.
%
%       USAGE:
%         V=hopfield(U,T,C,Tf,sigtype,maxval,lambda)

% Set default values if necessary
if nargin<7
  lambda=1;
end
if nargin<6
  maxval=1;
end

% Scale T and C by lambda
T=lambda*T;
C=lambda*C;

if sigtype==3                                    % Linear Neuron
  [Phi,Del]=c2d(T,C,Tf);                         % Discretize net
  V1=Phi*V0+Del;                                 % Iterate one step
else

if sigtype==4                                    % Saturating Neuron
  [Phi,Del]=c2d(T,C,Tf);                         % Discretize net
  V1=hard_lim(Phi*V0+Del,-maxval,maxval);      % Iterate and limit output
else

% Sigmoidal or tanH Neurons
% Routine based upon a modified version of MATLAB function ODE45.M

% The Fehlberg coefficients:
alpha = [1/4  3/8  12/13  1  1/2]';
beta  = [ [     1       0       0      0       0     0]/4
          [     3       9       0      0       0     0]/32
          [  1932   -7200    7296      0       0     0]/2197
          [  8341  -32832   29440   -845       0     0]/4104
          [ -6080   41040  -28352   9295   -5643     0]/20520 ]';
gamma = [ [902880  0  3953664  3855735  -1371249  277020]/7618050
          [ -2090  0    22528    21970    -15048  -27360]/752400 ]';
pow = 1/5;
```

```
trace = 0;
tol = 1.e-6;

% Initialization
t0=0;
tfinal=Tf;
y0=V0;
t = t0;
hmax = (tfinal - t)/5;
hmin = (tfinal - t)/20000;
h = (tfinal - t)/100;
y = y0(:);
f = y*zeros(1,6);
tout = t;
yout = y.';
tau = tol * max(norm(y, 'inf'), 1);

% The main loop
   while (t < tfinal) & (h >= hmin)
      if t + h > tfinal, h = tfinal - t; end

      % Compute the slopes
      % Call to neuron function sigmoid substituted here for
      %         nonlinear function
      f(:,1)=T*maxval*sigmoid(y/maxval,sigtype)+C;
      for j = 1:5
         f(:,j+1)=T*maxval*sigmoid((y+h*f*beta(:,j))/maxval,sigtype)+C;
      end

      % Estimate the error and the acceptable error
      delta = norm(h*f*gamma(:,2),'inf');
      tau = tol*max(norm(y,'inf'),1.0);

      % Update the solution only if the error is acceptable
      if delta <= tau
         t = t + h;
         y = y + h*f*gamma(:,1);
         tout = [tout; t];
         yout = [yout; y.'];
      end

      % Update the step size
      if delta ~= 0.0
         h = min(hmax, 0.8*h*(tau/delta)^pow);
      end
   end;

   if (t < tfinal)
      disp('SINGULARITY LIKELY.')
      t
   end
   V1=maxval*sigmoid(yout(length(tout),:)/maxval,sigtype)'
end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% HOPINIT.M %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% HOPINIT
%       This is the initialization script file for the direct
%       adaptive control Hopfield net problem files.
%       Variable set in this file include:
%
%       Ts - The sampling time
%       Tf - The final time
%       ZOH - the number of time steps in the zero order hold
%       sigtype - The neuron transfer characteristic (see SIGMOID.M)
%       lambda - The Hopfield net weight matrix gain
%       alpha - The pole of the filter for the T and C matrices in r/s
%       maxval - The maximum value the neurons are allowed to attain.
%               This is used for the saturating neurons
%       a - System denominator discrete time polynomial
%       b - System numerator discrete time polynomial
%       v - Reference signal
%
%       R. Scott Starsman 11-25-91

Ts=.1;                          % Sampling time
Tf=100;                         % Final time
t=0:Ts:Tf;
its=length(t);                  % # of iterations
ZOH=10;                         % Length in Ts of ZOH
sigtype=4;                      % Sigmoid type
lambda=1;                       % T and C matrix gain
alpha=1;                        % Filter pole for T and C matrices
maxval=inf;
rand('normal');

% Set up the test system (the system to be controlled is defined here)

% The depth model of the AUV
a1=.07;
b1=.04;
v=3;                            % Forward speed
K=1;
Zd=1;                           % Initial depth error

% State space model of the system
A=[-a1 -b1 0;1 0 0;0 -v*b1 0];
B=[1;0;0];
C=[0 0 1];
D=0;
x0=[0;0;Zd];                    % Initial conditions
[bc,ac]=ss2tf(A,B,C,D,1);       % Continuous time TF model
[Phi,Del]=c2d(a,b,Ts);          % Discrete time SS model
[b,a]=ss2tf(Phi,Del,c,d,1);     % Discrete time TF model

% Condition a and b
a=fliplr(a(2:length(a)));
b=fliplr(rlz(b));

% Set system order and  number of zeros
n=length(a);                    % System order
mc=length(rlz(bc))-1;           % Number of continuous time zeros
md=length(b)-1;                 % Number of discrete time zeros

% Set the number of controller parameters
numparms=2*n+1;
```

```
% Initialize system
y=Zd*ones(its,1);
u=[zeros(2*n-1,1);0.1];
sigmay=0;                       % Set measurement noise strength
noise=sigmay*rand(t');
T=zeros(numparms,numparms);
C=zeros(numparms,1);
clear theta
theta(:,1)=.1*ones(numparms,1);

% Determine the reference signal
%
% unit step
%v=ones(t);
%
% sin wave
%v=sin(t);
%
% Zero
%v=zeros(t);
%
% Whit Noise
%v=rand(t);
%
% Square wave
v=sign(sin(.2*t));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% HOPPROB4.M %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% HOPPROB4.M
%       This script file simulates the direct adaptive
%       Hopfield net control of a system.  The initialization
%       routine HOPINIT must be run prior to running this file.
%
%       R. Scott Starsman 11-18-91

rand('normal')
% Initialize copies of T and C for use in the ZOH
T1=T; C1=C;
t1=clock;                               % Set a timer

pstar=butterw(n-mc,1);                  % Determine the reference model
q=butterw(n,2);                         % Determine the observer
pstarq=conv(pstar,q);
[h1,k1,g1]=find_hk(ac,bc,pstar,q);      % Find the actual controller

% Determine th model's response to the input signal
model=lsim(1,pstar,v(1:length(t))*pstar(length(pstar)),t);

% Filter for derivatives of y and u
[Af,Bf,Cf,Df]=tf2ss(1,pstarq);
Cf=eye(Af);
Df=zeros(Bf);
[Phif,Delf]=c2d(Af,Bf,Ts);
ybar=zeros(2*n-mc,1);                   % Initialize the filters
ubar=zeros(2*n-mc,1);

% Filter for Calculating u
Aqs=[-q(2:n+1);eye(n-1) zeros(n-1,1)];
Aq=blokdiag(Aqs);
Bqs=[1;zeros(n-1,1)];
Bq=blokdiag(Bqs);
[Phiq,Delq]=c2d(Aq,Bq,Ts);
xq=zeros(2*n,1);                        % Initialize the filter

% Filter for the T and C matrices
pole=alpha;
A=-pole;
B=pole;
[PhiTC,DelTC]=c2d(A,B,Ts);

% This loop is to initialize the filters and the system
for k=n+1:2*n
  ybar=Phif*ybar+Delf*y(k-1);
  ubar=Phif*ubar+Delf*u(k-1);
  xq=Phiq*xq+Delq*[y(k-1); u(k-1)];
  u(k)=theta(:,1)'*[xq;v(k)];
  y(k)=-a*y(k-n:k-1)+b*u(k-md-1:k-1);
end

% Simulate the Hopfield net direct adaptive controller
for k=2*n+1:its
  y(k)=-a*y(k-n:k-1)+b*u(k-md-1:k-1);   % Iterate the system

  % Prepare phi(t) and s(t)
  ybar=Phif*ybar+Delf*y(k-1);           % Update the filtered versions
  ubar=Phif*ubar+Delf*u(k-1);           % of y(t) and u(t) and derivs
  phi=[ybar(n-mc+1:2*n-mc);ubar(n-mc+1:2*n-mc);y(k)];   % Form phi(t)
  s=q*ubar(n-mc:2*n-mc);                % Form s(t)
  % Determine T and C and apply ZOH
  T1=hard_lim(-DelTC*phi*phi'+PhiTC*T1,-maxval,maxval); % Update T
  C1=hard_lim(DelTC*phi*s+PhiTC*C1,-maxval,maxval);     % Update C
  if (k<20) ¦ (rem(k,ZOH)==0) T=T1; C=C1; end           % Apply ZOH

  % Iterate the Hopfield net
  theta(:,k-2*n+1)=hopfield(theta(:,k-2*n),T,C,Ts,lambda,maxval,sigtype);

  % Limit the estimate of gp from falling below g1/5
  if theta(numparms,k-2*n+1)>g1/5 theta(numparms,k-2*n+1)=g1/5; end

  % Filter y(t) and u(t) for calculating the control signal
  xq=Phiq*xq+Delq*[y(k-1); u(k-1)];
  u(k)=theta(:,k-2*n+1)'*[xq;v(k)*pstar(length(pstar))];

end

etime(clock,t1)                          % Stop the timer
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% PEND.M %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PEND.M
%       This is the MATLAB program for the Hopfield net control of
%       an inverted pendulum.
%
%       R. Scott Starsman 11-13-91

Ts=.02;                         % Sampling time
Tf=100;                         % Finish time
t=0:Ts:Tf;
its=length(t);                  % Number of iterations
ZOH=1;                          % Set the ZOH length
ZOHstart=500;                   % Set the iteration # to start ZOH
sigtype=4;                      % Transfer characteristic type
lambda=160000000000;            % Gain for T and C
pole=1;                         % Pole of filter for T and C
maxval=inf;
rand('normal');

b=1;                            % Pendulum damping
m=.1;                           % Pendulum mass
l=1;                            % Pendulum length
g=9.8;                          % gravity

n=2;                            % Estimated system order
mc=0;                           % Estimated continuous system zeros

numparms=2*n+1;                 % Number of parameters to estimate

% System initial conditions
y=.01*ones(its,1);              % y
yd=zeros(y);                    % y dot
u=[zeros(2*n-1,1);0.1];
sigmay=0;
sigmau=0;
noise=sigmay*rand(t');
T=zeros(numparms,numparms);
C=zeros(numparms,1);
theta(:,1)=.1*ones(numparms,1);

% This is the reference signal - square wave
v=.1*sign(sin(.5*t))+.0*rand(t);

% Hopfield Problem for upside down pendulum

t1=clock;                       % Set a timer
modfreq=1;                      % Set frequency of the reference model
pstar=butterw(n-mc,modfreq);    % The reference model
q=butterw(n,4*modfreq);         % The observer
pstarq=conv(pstar,q);

% Estimate the value of h(s), k(s), and gp
[h1,k1,g1]=find_hk([1 b/m/l^2 -g/l],1/m/l^2,pstar,q);

model=lsim(1,pstar,v(1:length(t)),t);   % The reference model output

% Filter for derivatives of y and u
[Af,Bf,Cf,Df]=tf2ss(1,pstarq);
Cf=pstarq(length(pstarq))*eye(Af);
Df=zeros(Bf);
[Phif,Delf]=c2d(Af,Bf,Ts);
```

```
ybar=zeros(2*n-mc,1);
ubar=zeros(2*n-mc,1);

% Filter for Calculating u
Aqs=[-q(2:n+1);eye(n-1) zeros(n-1,1)];
Aq=blokdiag(Aqs);
Bqs=[1;zeros(n-1,1)];
Bq=blokdiag(Bqs);
Cq=q(length(q))*eye(Aq);
[Phiq,Delq]=c2d(Aq,Bq,Ts);
xq=zeros(2*n,1);

% Filter for the T and C matrices
A=-pole;
B=pole;
[PhiTC,DelTC]=c2d(A,B,Ts);

% Initialize the system and filters
for k=n+1:2*n
  % The pendulum
  yd(k)=yd(k-1)-b/m/l^2*yd(k-1)*Ts+g/l*sin(y(k-1))*Ts;
  y(k)=y(k-1)+yd(k-1)*Ts;

  ybar=Phif*ybar+Delf*y(k-1);
  ubar=Phif*ubar+Delf*u(k-1);
  phi=[ybar(n-mc+1:2*n-mc);ubar(n-mc+1:2*n-mc);y(k-1)];
  s=q*ubar(n-mc:2*n-mc);
  T=hard_lim(-DelTC*phi*phi'+PhiTC*T,-maxval,maxval);
  C=hard_lim(DelTC*phi*s+PhiTC*C,-maxval,maxval);
  xq=(Phiq*xq+Delq*[y(k-1); u(k-1)]);
  u(k)=theta(:,1)'*[xq;v(k)];
end

% Simulate the system
for k=2*n+1:its
  % The pendulum
  yd(k)=yd(k-1)-b/m/l^2*yd(k-1)*Ts+g/l*sin(y(k-1))*Ts+u(k-1)/m/l^2*Ts;
  y(k)=y(k-1)+yd(k-1)*Ts;

  % Determine the phi vector and s(t)
  ybar=Phif*ybar+Delf*y(k-1);
  ubar=Phif*ubar+Delf*u(k-1);
  phi=[ybar(n-mc+1:2*n-mc);ubar(n-mc+1:2*n-mc);y(k-1)];
  s=q*ubar(n-mc:2*n-mc);

  % Update T and C
  T=hard_lim(-DelTC*phi*phi'+PhiTC*T,-maxval,maxval);
  C=hard_lim(DelTC*phi*s+PhiTC*C,-maxval,maxval);

  % Iterate the Hopfield net
  theta(:,k-2*n+1)=hopfield(theta(:,k-2*n),T,C,Ts,sigtype,maxval,lambda);

  % Prevent the estimate of gp from falling too low
  if theta(numparms,k-2*n+1)<gl/2 theta(numparms,k-2*n+1)=gl/2; end

  % Filter y(t) and u(t) for the output calculations
  xq=(Phiq*xq+Delq*[y(k-1); u(k-1)]);

  % Determine the system control signal
  u(k)=hard_lim(theta(:,k-2*n+1)'*[xq;v(k)],-30,30);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% GET_MOD.M %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GET_MOD.M
%       This file is used to determine a set of system models for
%       the input/output data held in the vectors y and u
%       respectively.  These vectors are assumed to exist upon entry
%       into this routine.  The routine finds a set of continuous time
%       transfer function models for varying plant order and numerator
%       order.
%
%       R. Scott Starsman 11-25-91

% Clear these variables for later use
nt=[]; dt=[]; results=[];

% Nested loop to determine the system order and number of zeros
for n=1:5                       % First to fifth order
  for m=0:n-1                   % # of zeros less than number of poles
    % Find the model
    [num,den,theta]=find_mod(y,u',n,m,Ts);
    y2=lsim(num,den,u,t);       % The model's response to the input
    e=y-y2;                     % Generate an error vector
    nt=[nt;zeros(1,6-m) num];   % Save the model numerator
    dt=[dt;zeros(1,6-n) den];   % Save the model denominator

    % Plot the actual data vs the model
    clg
    subplot(211)
    plot(t,y,t,y2,t,e)
    subplot(212)
    % Plot the convergence of the model parameters
    plot(t(1:length(theta)),theta')

    % Save the model order, # of zeros, and summed absolute error
    results=[results;n m sum(abs(e))];
  end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% FIND_MOD.M %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [num,den,theta,P]=find_mod(y,u,n,m,Ts)
% FIND_MOD.M - USAGE: [num,den,theta,P]=find_mod(y,u,n,m,Ts)
%       This routine returns a continuous time transfer function
%       model of order n with m zeros for the input/output data y
%       and u.  The time interval between data points is given by Ts.
%
%       y - y is a column vector of system output data
%       u - u is a column vector of the system input
%       n - n is the model system order
%       m - m is the number of modeled zeros
%       Ts - Ts is the time interval between data points
%
%       num - num is the numerator polynomial of the modeled TF
%       den - den is the denominator polynomial of the modeled TF
%       theta - theta is the vector of parameters over time.  This is
%               useful for examining the convergence of the parameters
%       P - P is the error covariance matrix.
%
%       R. Scott Starsman 11-15-91

numparms=n+m+1;                         % Number of parameters to find

theta=zeros(numparms,1);                % Initialize theta

% Estimate n-1 derivatives of y and store them all in yprime
yprime=y;
for k=2:n
  yprime=[[diff(yprime(:,1))/Ts;0] yprime];
end
ydot=[diff(yprime(:,1))/Ts;0];

% Estimate m derivatives of u and store them in uprime
uprime=u;
for k=2:m+1;
  uprime=[[diff(uprime(:,1))/Ts;0] uprime];
end

% Set up a matrix of y and its derivatives and u and its derivatives
phi=[-yprime uprime];

% Initialize error covariance matrix
P=1e8*eye(numparms,numparms);

% Estimate system parameters using RLS
for k=1:length(y)-numparms
  den=1+phi(k,:)*P*phi(k,:)';
  theta(:,k+1)=theta(:,k)+P*phi(k,:)'*(ydot(k)-phi(k,:)*theta(:,k))/den;
  P=P-P*phi(k,:)'*phi(k,:)*P/den;
end

% Return the numerator and denominator of the system
den=[1 theta(1:n,length(theta))'];
num=theta(n+1:numparms,length(theta))';
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% AUVPATH1.M %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% AUVPATH1.M
%       This file is the driver for the direct adaptive Hopfield
%       net control of the AUV.  This file uses the Kanayama
%       path planning algorithm to generate state reference
%       signals for the course rate, depth rate, and surge of the
%       AUV.  These state control signals are passed to the Hopfield
%       net controller for actuation of the control surfaces.

% Initialize the system
dt=.25;                         % Sample time
Tf=300;                         % Final time
t=0:dt:Tf;
its=length(t);                  % Number of iterations
ZOH=1;                          % ZOH time in numbers of time steps
lambda=1;                       % T and C matrix gain
sigtype=3;                      % Sigmoid type
maxval=inf;                     % Neuron output limit
rand('normal')

% Since three states need to be controlled, three Hopfield
% nets are required.  Variables dealing with the course rate
% are appended with 'c', with the depth rate are appended 'd',
% and with the surge are appended 's'

% Set the system order and the size of the Hopfield net
nc=1; mc=0; numparmc=2*nc+1;
ns=1; ms=0; numparms=2*ns+1;
nd=1; md=0; numparmd=2*nd+1;

% Initialize T, C, the input, the output, and the auv position
Tc=zeros(numparmc,numparmc); Cc=zeros(numparmc,1); gc=-5.6;
Ts=zeros(numparms,numparms); Cs=zeros(numparms,1); gs=855.7;
Td=zeros(numparmd,numparmd); Cd=zeros(numparmd,1); gd=3.973;
yc=0*ones(t); uc=zeros(yc);
ys=2*ones(t); us=400*ones(t);
yd=0*ones(t); ud=zeros(yd);
d0=0;                           % Initial depth
auvx=zeros(t); auvy=zeros(t); auvd=d0*ones(t);
% Initial AUV state
x0=[ys(1);0;yd(1);0;0;yc(1);0;0;d0;0;0;0];

t1=clock;                       % Start a timer

% The first order system models for the three subsystems
numc=-0.167;denc=[1 0.5];
nums=0.00065;dens=[1 0.128];
numd=0.183;dend=[1 0.088];
% The system reference models and observers
pstarc=butterw(nc-mc,.5); qc=butterw(nc,1); pstarqc=conv(pstarc,qc);
pstars=butterw(ns-ms,.2); qs=butterw(ns,.8); pstarqs=conv(pstars,qs);
pstard=butterw(nd-md,.5); qd=butterw(nd,1); pstarqd=conv(pstard,qd);
% Estimate the control parameters from the system models
[hc,kc,gc]=find_hk(denc,numc,pstarc,qc);thetac=[hc;kc;gc];
[hs,ks,gs]=find_hk(dens,nums,pstars,qs);thetas=[hs;ks;gs];
[hd,kd,gd]=find_hk(dend,numd,pstard,qd);thetad=[hd;kd;gd];

% Parameters and IC's for path follower
% Kanayama path follower
Kx=.1;
Ky=.1;
```

```
Ktheta=sqrt(4*Ky);                  % Critically damped system in CRE
Kd=.5;
R0=0;                               % Following range in ft
vref=2;                             % Initial reference speed
wref=0;                             % Initial reference course rate
targddot=0;                         % Initial reference depth rate
% Initial reference position in X, Y, Z
targxyd=[20 20 20 20 20;
         20 20 20 20 20;
         20 20 20 20 20];
thetaref=0;                         % Initial reference heading
vc=wref*ones(yc); vs=vref*ones(ys); vd=targddot*ones(yd);

% Filter for derivatives of y and u
[Afc,Bfc,Cfc,Dfc]=tf2ss(1,pstarqc);
[Phifc,Delfc]=c2d(Afc,Bfc,dt);
[Afs,Bfs,Cfs,Dfs]=tf2ss(1,pstarqs);
[Phifs,Delfs]=c2d(Afs,Bfs,dt);
[Afd,Bfd,Cfd,Dfd]=tf2ss(1,pstarqd);
[Phifd,Delfd]=c2d(Afd,Bfd,dt);
ybarc=zeros(2*nc-mc,1); ubarc=zeros(2*nc-mc,1);
ybars=zeros(2*ns-ms,1); ubars=zeros(2*ns-ms,1);
ybard=zeros(2*nd-md,1); ubard=zeros(2*nd-md,1);

% Filter for Calculating u
Aqs=[-qc(2:nc+1);eye(nc-1) zeros(nc-1,1)];
Aqc=blokdiag(Aqs);
Bqs=[1;zeros(nc-1,1)];
Bqc=blokdiag(Bqs);
[Phiqc,Delqc]=c2d(Aqc,Bqc,dt);
Aqs=[-qs(2:ns+1);eye(ns-1) zeros(ns-1,1)];
Aqs=blokdiag(Aqs);
Bqs=[1;zeros(ns-1,1)];
Bqs=blokdiag(Bqs);
[Phiqs,Delqs]=c2d(Aqs,Bqs,dt);
Aqs=[-qd(2:nd+1);eye(nd-1) zeros(nd-1,1)];
Aqd=blokdiag(Aqs);
Bqs=[1;zeros(nd-1,1)];
Bqd=blokdiag(Bqs);
[Phiqd,Delqd]=c2d(Aqd,Bqd,dt);
xqc=zeros(2*nc,1);
xqs=zeros(2*ns,1);
xqd=zeros(2*nd,1);


% Filter for the T and C matrices
pole=1;
A=-pole;
B=pole;
[PhiTC,DelTC]=c2d(A,B,dt);

% Determine necessary length of initialization
n=max([nd;nc;ns]);

% Initialize the filters and the system
for k=n+1:2*n
  ybarc=Phifc*ybarc+Delfc*yc(k-1);
  ubarc=Phifc*ubarc+Delfc*uc(k-1);
  xqc=Phiqc*xqc+Delqc*[yc(k-1); uc(k-1)];
  uc(k)=hard_lim(thetac(:,1)'*[xqc;vc(k)],-.4,.4);
  ybars=Phifs*ybars+Delfs*ys(k-1);
  ubars=Phifs*ubars+Delfs*us(k-1);
  xqs=Phiqs*xqs+Delqs*[ys(k-1); us(k-1)];
  us(k)=hard_lim(thetas(:,1)'*[xqs;vs(k)],0,750);
  ybard=Phifd*ybard+Delfd*yd(k-1);
  ubard=Phifd*ubard+Delfd*ud(k-1);
  xqd=Phiqd*xqd+Delqd*[yd(k-1); ud(k-1)];
  ud(k)=hard_lim(thetad(:,1)'*[xqd;vd(k)],-.4,.4);
  inputs=[uc(k); -uc(k); ud(k); -ud(k); us(k)];
  x0=auv2(x0,inputs,dt);
  yc(k)=x0(6); ys(k)=x0(1); yd(k)=(x0(9)-d0)/dt;
  d0=x0(9)
  auvx(k)=x0(7); auvy(k)=x0(8); auvd(k)=x0(9);
  thetaref=thetaref+wref*dt;

targxyd(:,k+1)=targxyd(:,k)+[cos(thetaref)*vref;sin(thetaref)*vref;targd
dot]*dt;
end

% Simulate the AUV
for k=2*n+1:its
  x0=auv2(x0,inputs,dt);                % Iterate the AUV one step
  % Pick out the course rate, depth rate, and surge
  yc(k)=x0(6); ys(k)=x0(1); yd(k)=(x0(9)-d0)/dt;
  d0=x0(9);                             % Keep track of the old depth
  % Update the AUV's position
  auvx(k)=x0(7); auvy(k)=x0(8); auvd(k)=x0(9);

  % Path follower definitions
  if t(k)==70 vref=2; targxyd(3,k)=25; thetaref=pi/2; end
  if t(k)==95 vref=2; thetaref=pi; end
  if t(k)==105 vref=3; targxyd(3,k)=20; end
  if t(k)==110 vref=2; end
  if t(k)==200 vref=3; thetaref=-pi/2; end
  if t(k)==250 vref=2; thetaref=0; end

  % Update the path follower
  % Range from AUV to reference point
  R(k)=sqrt((targxyd(1,k)-auvx(k))^2+(targxyd(2,k)-auvy(k))^2);

  % Determine the error posture
  temp=[cos(x0(12)) sin(x0(12));-sin(x0(12)) cos(x0(12))];
  xyfollow=targxyd(1:2,k)-R0*[cos(thetaref);sin(thetaref)];
  e=temp*(xyfollow-[auvx(k);auvy(k)]);
  thetae=thetaref-x0(12);

  % Determine the command signals
  vc(k)=wref+vref*(Ky*e(2)+Ktheta*sin(thetae));
  vs(k)=hard_lim(vref*cos(thetae)+Kx*e(1),0,5);
  vd(k)=targddot+Kd*(targxyd(3,k)-x0(9));

  % Update the reference points posture
  thetaref=thetaref+wref*dt;
  deltarg=[cos(thetaref)*vref;sin(thetaref)*vref;targddot]*dt;
  targxyd(:,k+1)=targxyd(:,k)+deltarg;

  % Course Rate controller
  % Determine phi and s
  ybarc=Phifc*ybarc+Delfc*yc(k-1);
  ubarc=Phifc*ubarc+Delfc*uc(k-1);
  phi=[ybarc(nc-mc+1:2*nc-mc);ubarc(nc-mc+1:2*nc-mc);yc(k)];
  s=qc*ubarc(nc-mc:2*nc-mc);
  % Determine T and C and iterate the Hopfield net
  Tc=hard_lim(-DelTC*phi*phi'+PhiTC*Tc,-maxval,maxval);
  Cc=hard_lim(DelTC*phi*s+PhiTC*Cc,-maxval,maxval);
  thetac(:,k-2*n+1)=...
       hopfield(thetac(:,k-2*n),Tc,Cc,dt,sigtype,maxval,lambda);
  if thetac(numparmc,k-2*n+1)>gc/5 thetac(numparmc,k-2*n+1)=gc/5; end

  % Determine the system control signal for the course rate
  xqc=Phiqc*xqc+Delqc*[yc(k-1); uc(k-1)];
  uc(k)=thetac(:,k-2*n+1)'*[xqc;vc(k)*pstarc(length(pstarc))];
  uc(k)=hard_lim(uc(k),-.4,.4);

  % Speed Controller
  % Determine phi and s
  ybars=Phifs*ybars+Delfs*ys(k-1);
  ubars=Phifs*ubars+Delfs*us(k-1);
  phi=[ybars(ns-ms+1:2*ns-ms);ubars(ns-ms+1:2*ns-ms);ys(k)];
  s=qs*ubars(ns-ms:2*ns-ms);

  % Determine T and C and iterate the Hopfield net
  Ts=hard_lim(-DelTC*phi*phi'+PhiTC*Ts,-maxval,maxval);
  Cs=hard_lim(DelTC*phi*s+PhiTC*Cs,-maxval,maxval);
  thetas(:,k-2*n+1)=...
       hopfield(thetas(:,k-2*n),Ts,Cs,dt,sigtype,maxval,lambda);
  if thetas(numparms,k-2*n+1)<gs/5 thetas(numparms,k-2*n+1)=gs/5; end

  % Determine the control signal for the surge
  xqs=Phiqs*xqs+Delqs*[ys(k-1); us(k-1)];
  us(k)=thetas(:,k-2*n+1)'*[xqs;vs(k)*pstars(length(pstars))];
  us(k)=hard_lim(us(k),110,750);

  % Depth Controller
  % Determine phi and s
  ybard=Phifd*ybard+Delfd*yd(k-1);
  ubard=Phifd*ubard+Delfd*ud(k-1);
  phi=[ybard(nd-md+1:2*nd-md);ubard(nd-md+1:2*nd-md);yd(k)];
  s=qd*ubard(nd-md:2*nd-md);

  % Determine T and C and iterate the Hopfield net
  Td=hard_lim(-DelTC*phi*phi'+PhiTC*Td,-maxval,maxval);
  Cd=hard_lim(DelTC*phi*s+PhiTC*Cd,-maxval,maxval);
  thetad(:,k-2*n+1)=...
       hopfield(thetad(:,k-2*n),Td,Cd,dt,sigtype,maxval,lambda);
  if thetad(numparmd,k-2*n+1)<gd/5 thetad(numparmd,k-2*n+1)=gd/5; end

  % Determine the control signal for the depth rate
  xqd=Phiqd*xqd+Delqd*[yd(k-1); ud(k-1)];
  ud(k)=thetad(:,k-2*n+1)'*[xqd;vd(k)*pstard(length(pstard))];
  ud(k)=hard_lim(ud(k),-.4,.4);

  % Form the input vector for the AUV model
  inputs=[uc(k); -uc(k); ud(k); -ud(k); us(k)];
end
etime(clock,t1)                 % Stop the timer
```

# APPENDIX B.  TUTSIM CODE

```
PROFESSIONAL VERSION OF TUTSIM

Model File: hopfield
Date: 11/ 23 / 1991
Time: 22: 0
Timing: 0.0100000   , DELTA ; ,100.0000,   ' RANGE
PlotBlocks and Scales :
Format :
     BlockNo , plot-MINimum, Plot-MAXimum;  Comment                  Horz: 0 ,
0,0000 , 100.0000 ; Time
Y1:          26 , -1,5000 ,  1.5000  ;  y
Y2:          27 , -1.5000 ,  1.5000  ;  v
Y3:          20 , -1,5000 ,  1.5000  ;  thetal
Y4:          21 , -1.5000 ,  1.5000  ;  theta2


2.0000       1    GAI    2
0.0000       2    INT    3        ; y HAT
             3    SUM   -1 30     ; y HAT dot
             4    MUL    3  3
             5    MUL   2   3
             6    MUL   3  25
             7    MUL   2   2
             8    MUL   2  25
1.0000       9    FIO  -4          ; Tll
0.0100000
-0.1000000
1.0000       10   FIO   -5        ; T12 or T21
0.0100000
-0 1000000
1.0000       11   FIO   -7        ; T22
0.0100000
-0.1000000
1.0000       12   FIO    8        ; C2
0.0100000
0.1000000
1.0000       13   FIO    6        ; Cl
0.0100000
0.1000000
             14   MUL  20   9
             15   MUL  20 10
             16   MUL  21 10
             17   MUL  21 11
             18   SUM  13 14 16
             19   SUM  12 15 17
0.1000000    20   INT  18          ; thetal
0.1000000    21   INT  19          ; theta2
             22   MUL  20   29
             23   MUL  21   26
             24   SUM  22   23    ; u
1.0000       25   FIO  24          ; u HAT
0.5000000
0.0000
```

```
1.5000          26   FIO  28          ; y
0.2000000
0.0000
                27   SGN  31          ; v
1.0000          28   GAI  24
                29   SUM -26  27
                30   GAI  26
0.0500000       31   FRQ
1.0000
```

# APPENDIX C.  SPICE SOFTWARE

```
FIRST ORDER HOPFIELD NET CONTROLLER

* IMPORTANT NODES:
* 1 -> y
* 2 -> yHAT
* 3 -> yHATDOT
* ¦ 9  10 ¦ -> ¦ T11  T12 ¦
* ¦ 10 11 ¦ -> ¦ T21  T22 ¦
* ¦ 12 ¦ -> ¦ C1 ¦
* ¦ 13 ¦ -> ¦ C2 ¦
* 20 -> THETA1
* 21 -> THETA2
* 24 -> u
* 25 -> uHAT
* 27 -> v

X1 24 101 102 1 SYSTEM
X2 1 101 102 2 3 YFILTER
X3 3 3 101 102 4 MULT
X4 2 3 101 102 5 MULT
X5 2 2 101 102 6 MULT
X6 3 25 101 102 7 MULT
X7 2 25 101 102 8 MULT
X8 4 101 102 9 TFILTER
X9 5 101 102 10 TFILTER
X10 6 101 102 11 TFILTER
X11 7 101 102 12 CFILTER
X12 8 101 102 13 CFILTER
X13 9 20 101 102 14 MULT
X14 10 20 101 102 15 MULT
X15 10 21 101 102 16 MULT
X16 11 21 101 102 17 MULT
X17 12 14 16 101 102 18 SUMMER3
X18 13 15 17 101 102 19 SUMMER3
X19 18 101 102 20 INTEG
X20 19 101 102 21 INTEG
X21 20 26 101 102 22 MULT
X22 1 21 101 102 23 MULT
X23 22 23 101 102 24 SUMMER2
X24 28 101 102 24 INVERTER
X25 24 101 102 25 UFILTER
X26 27 1 101 102 26 DIFF
VV 27 0 PULSE (-5 5 .1 0 0 4 8)
VDD 101 0 DC 12V
VSS 102 0 DC -12V
.TRAN 100MS 20S
.PLOT TRAN V(1) V(27)
```

```
.SUBCKT SYSTEM 3    1    2    4
*              u    Vdd  Vss  y
* FIRST ORDER SYSTEM --> y/u=G/(Ts+1)
*  G=(R1+R2)/R1;    T=R3*C1
R1 6 0 100K
R2 4 6 50K
R3 3 5 200K
C1 5 0 1U
X1 5 6 1 2 4 MOPAMP
.ENDS SYSTEM

.SUBCKT CFILTER 3    1    2    4
*               Cin  Vdd  Vss  Cout
* FILTER FOR C VECTOR --> Cout/Cin=G/(Ts+1)
*  G=1;    T=R3*C1
R1 4 0 100K
R3 3 5 100K
C1 5 0 1U
X1 5 4 1 2 4 MOPAMP
.ENDS CFILTER

.SUBCKT TFILTER 3    1    2    4
*               Tin  Vdd  Vss  Tout
* FILTER FOR T MATRIX --> Tout/Tin=-G/(Ts+1)
*  G=1;   T=R1*C1;   R1=R2=R3;   R4=2*R1
R1 3 7 100K
R2 6 7 100K
R3 4 6 200K
R4 5 0 200K
C1 7 0 1U
C2 4 6 .5U
X1 5 6 1 2 4 MOPAMP
.ENDS TFILTER

.SUBCKT YFILTER 11   1    2    8    3
*               y    Vdd  Vss  yHAT yHATDOT
*  FILTER  TO  DERIVE  yHAT  AND  yHATDOT  -->  yHAT/y=1/(Ts+1);
yHATDOT/y=s/(Ts+1)
*  T=R2/R1=R4/R3
R1 11 12 10K
R2 12 0 20K
R3 9 8 10K
R4 9 3 20K
X1 12 9 1 2 3 MOPAMP
X2 3 1 2 5 INTEG
X3 5 1 2 8 INVERTER
.ENDS YFILTER

.SUBCKT UFILTER 3    1    2    4
*               Uin  Vdd  Vss  Uout
* FILTER FOR C VECTOR --> Uout/Uin=G/(Ts+1)
*  G=1;   T=R3*C1
R1 4 0 500K
R3 3 5 500K
C1 5 0 1U
X1 5 4 1 2 4 MOPAMP
.ENDS UFILTER
```

```
.SUBCKT INVERTER 3    1    2    4
*                Vin  Vdd  Vss  Vout
* INVERTER --> Vout=-G*Vin
*  G=R2/R1;   R3=R1¦¦R2
R1 3 6 100K
R2 4 6 100K
R3 5 0 50K
X1 5 6 1 2 4 MOPAMP
.ENDS INVERTER

.SUBCKT DIFF 3    4    1    2    5
*            X    Y    Vdd  Vss  X-Y
* DIFFERENCE CIRCUIT
R1 3 6 100K
R2 4 7 100K
R3 6 0 100K
R4 5 7 100K
X1 6 7 1 2 5 MOPAMP
.ENDS DIFF

.SUBCKT SUMMER2 3    4    1    2    5
*               X    Y    Vdd  Vss  -(X+Y)
* 2 INPUT SUMMER CIRCUIT
R1 3 7 100K
R2 4 7 100K
R3 5 7 100K
R4 6 0 33K
X1 6 7 1 2 5 MOPAMP
.ENDS SUMMER2

.SUBCKT SUMMER3 3    4    5    1    2    6
*               X    Y    Z    Vdd  Vss  -(X+Y+Z)
* 3 INPUT SUMMER CIRCUIT
R1 3 8 100K
R2 4 8 100K
R3 5 8 100K
R4 6 8 100K
R5 7 0 25K
X1 7 8 1 2 6 MOPAMP
.ENDS SUMMER3

.SUBCKT INTEG 3    1    2    4
*             Vin  Vdd  Vss  Vout
* INTEGRATOR --> Vout/Vin=G/s
*  G=1/(R1*C1);   R2=R1
R1 3 5 1MEG
R2 6 0 1MEG
C1 4 5 1U
X1 6 5 1 2 4 MOPAMP
.ENDS INTEG

.SUBCKT MULT 3    4    1    2    12
*            X    Y    Vdd  Vss  Vout
*ANALOG VOLTAGE MULTIPLIER FROM HONG & MELCHIOR (ELEC LTTRS 6 JUN 85)
R1 3 5 5K
R2 0 5 5K
R3 4 6 5K
R4 0 6 5K
R5 4 7 5K
R6 3 7 5K
M1 1 5 8 1 CMOSP L=32U W=16U
```

```
M2 1 6 8 1 CMOSP L=32U W=16U
M3 1 7 9 1 CMOSP L=32U W=16U
M4 1 0 9 1 CMOSP L=32U W=16U
X1 9 8 1 2 10 MOPAMP
R7 11 9 10K
R8 10 8 10K
E1 12 0 10 11 297
ROUT 12 0 1M
V2MINUS 11 0 DC -10
.ENDS MULT

.SUBCKT COPAMP3 3   4   1   2   5
*               V+  V-  Vdd Vss Vout
*CMOS OPAMP FROM MICROELECTRONIC CIRCUITS (SEDRA AND SMITH)
M1 9 4 6 1 CMOSP L=8U W=120U
M2 8 3 6 1 CMOSP L=8U W=120U
M3 9 9 2 2 CMOSN L=10U W=50U
M4 8 9 2 2 CMOSN L=10U W=50U
M5 6 7 1 1 CMOSP L=10U W=150U
M6 5 8 2 2 CMOSN L=10U W=100U
M7 5 7 1 1 CMOSP L=10U W=150U
M8 7 7 1 1 CMOSP L=10U W=150U
C1 8 10 10PF
R4 10 5 10K
M9 7 7 2 2 CMOSN L=250U W=5U
.ENDS COPAMP3

.SUBCKT MOPAMP 1   2   98   99   3
*              V+  V-  Vdd  Vss  Vout
RIN 1 2 6MEG
ROUT 4 3 75
E1 0 4 1 2 200K
* R1 AND R2 ARE USED TO MAKE THE OPAMP IDEAL MODEL PIN COMPATIBLE
* WITH THE CMOS OPAMP
R1 98 0 10M
R2 99 0 10M
.ENDS MOPAMP

.MODEL CMOSN NMOS LEVEL=2.00000 LD=0.6U TOX=8.5E-8
+NSUB=1E+16 VTO=1 JS=1E-6 UO=750.000 UEXP=.14 UCRIT=5E4 UTRA=0 PB=.7
+VMAX=5E4 XJ=1U CGBO=2E-10
+RSH=15 CGSO=4E-10 CGDO=4E-10 CJ=4E-04 MJ=2 CJSW=8E-10 MJSW=2
.MODEL CMOSP PMOS LEVEL=2.00000 LD=0.6U TOX=1E-7 NSUB=2E+15 VTO=-1 JS=1E-6
+UO=100.000 UEXP=.03 UCRIT=1E4 UTRA=0 PB=.7 VMAX=3E4 XJ=.9U CGBO=2E-10
+RSH=75 CGSO=4E-10 CGDO=4E-10 CJ=1.8E-04 MJ=2 CJSW=6E-10 MJSW=2
.END
```

# LIST OF REFERENCES

1. Schwartz, M. A., *Kalman Filtering for Adaptive Depth, Steering, and Roll Control of an Autonomous Underwater Vehicle (AUV)*, Master's Thesis, Naval Postgraduate School, Monterey, CA, March 1991.

2. Åström, K. and Wittenmark, B., *Computer Controlled Systems: Theory and Design*, Prentice-Hall, 1990.

3. Kailath, T., *Linear Systems*, Prentice-Hall, 1989.

4. Cristi, R., Notes for EC4360 (System Identification), Naval Postgraduate School, 1991 (unpublished).

5. Söderström, T. and Stoica, P., *System Identification*, Prentice-Hall, 1989.

6. NeuralWare Inc., *Neural Computing*, 1991.

7. Kosko, B., *Neural Networks and Fuzzy Systems*, Prentice-Hall, 1992.

8. Hecht-Nielsen, R., *Neurocomputing*, Addison-Wesley Publishing Co., 1990.

9. Sedra, A. and Smith, K., *Microelectronic Circuits*, Holt, Reinhart, and Winston, 1987.

10. The MathWorks Inc., *MATLAB Programmer's Manual*, 1986.

11. Warner, D., *Experimental Verification of a Computer Model and Enhanced Position Estimator for the NPS AUV II*, Master's Thesis, Naval Postgraduate School, Monterey, CA, December, 1991.

12. Miller, C., *An Application of Extended Kalman Filtering to a Model-Based Short-Range Navigator for an AUV*, Master's Thesis, Naval Postgraduate School, Monterey, CA, December, 1991.

13. Kanayama, Y., and others, "A Stable Tracking Control Method for an Autonomous Mobile Robot", paper presented at the IEEE International Conference on Robotics and Automation, Cincinnati, Ohio, 13-18 May, 1990.

14. Horn, D., *How to Design Op Amp Circuits with Projects and Experiments*, TAB Books Inc, 1984.

15. Analog Devices, *Analog Signal Processing Components*, Vol I, 1989.

16. Gregorian, R. and Temes, G. C., *Analog MOS Integrated Circuits of Signal Processing*, John Wiley and Sons, 1986.

17. Hong, Z. and Melchior, H., "Analogue Four Quadrant CMOS Multiplier with Resistors", *Electronics Letters*, v. 21, pp. 531-532, June 6, 1985.

18. Ogata, K., *Modern Control Engineering*, Prentice-Hall, 1970.

# INITIAL DISTRIBUTION LIST

| | | No. of Copies |
|---|---|---|
| 1. | Defense Technical Information Center<br>Cameron Station<br>Alexandria, Virginia, 22304-6145 | 2 |
| 2. | Library, Code 52<br>Naval Postgraduate School<br>Monterey, California, 93943-5002 | 2 |
| 3. | Chairman, Code EC<br>Department of Electrical and Computer Engineering<br>Naval Postgraduate School<br>Monterey, California, 93943 | 1 |
| 4. | Dr. A. J. Healey, Code ME<br>AUV Project<br>Department of Mechanical Engineering<br>Naval Postgraduate School<br>Monterey, California, 93943 | 1 |
| 5. | Dr. Roberto Cristi, Code EC/Cx<br>Department of Electrical and Computer Engineering<br>Naval Postgraduate School<br>Monterey, California, 93943 | 2 |
| 6. | Dr. Murali Tummala, Code EC/Tu<br>Department of Electrical and Computer Engineering<br>Naval Postgraduate School<br>Monterey, California, 93943 | 1 |
| 7. | Dr. Yutaka Kanayama, Code CS/Ka<br>Department of Computer Science<br>Naval Postgraduate School<br>Monterey, California, 93943 | 1 |
| 8. | Mr. Robert Wilson<br>Head, Systems Engineering Branch<br>David Taylor Research Center<br>Carderock, Bethesda, Maryland, 20084-5000 | 1 |
| 9. | Mr. Dan Steiger, Marine Systems Group<br>Naval Research Laboratory<br>Washington, D. C., 20032 | 1 |

10. Mr. Kirk Dye 1
Naval Coastal Systems Center
Panama City, Florida, 32407-5000

11. Technical Library 1
Naval Surface Warfare Center
Silver Spring, Maryland, 20901

12. RADM Evans, Code SEA-92 1
Naval Sea Systems Command
Washington, D. C., 20362

846-217

Thesis
S67775 Starsman
c.1 A Hopfield network
approach to direct
adaptive control of
nonlinear systems.

Thesis
S67775 Starsman
c.1 A Hopfield network
approach to direct
adaptive control of
nonlinear systems.

AMERICAN BINDERY
2
9-93

DUDLEY KNOX LIBRARY
3 2768 00036289 1